Volume 21

Pages 3717 - 3886

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

BEFORE THE HONORABLE WILLIAM H. ALSUP

```
ORACLE AMERICA, INC.,                 )
                                      )
             Plaintiff,               )
                                      )
  VS.                                 )  No. C 10-3561 WHA
                                      )
GOOGLE, INC.,                         )
                                      )
             Defendant.               )  San Francisco, California
______________________________________)  May 11, 2012
```

TRANSCRIPT OF JURY TRIAL PROCEEDINGS

APPEARANCES:

For Plaintiff:

MORRISON & FOERSTER
755 Page Mill Road
Palo Alto, California 94304
BY: **MICHAEL A. JACOBS, ESQUIRE**
KENNETH A. KUWAYTI, ESQUIRE
MARC DAVID PETERS, ESQUIRE
DANIEL P. MUINO, ESQUIRE

BOIES, SCHILLER & FLEXNER
333 Main Street
Armonk, New York 10504
BY: **DAVID BOIES, ESQUIRE**
ALANNA RUTHERFORD, ESQUIRE

(Appearances continued on next page)

Reported By: Katherine Powell Sullivan, RPR, CRR, CSR #5812
Debra L. Pas, RMR, CRR, CSR #11916
Official Reporters - U.S. District Court

APPEARANCES (CONTINUED):

For Plaintiff:

BOIES, SCHILLER & FLEXNER
1999 Harrison Street, Suite 900
Oakland, California 94612
BY: WILLIAM FRED NORTON, ESQUIRE
STEVEN C. HOLTZMAN, ESQUIRE

ORACLE AMERICA, INC.
500 Oracle Parkway
Redwood Shores, California 94065
BY: ANDREW C. TEMKIN, CORPORATE COUNSEL
DORIAN DALEY, GENERAL COUNSEL

For Defendant:

KEKER & VAN NEST
633 Battery Street
San Francisco, California 94111-1809
BY: ROBERT ADDY VAN NEST, ESQUIRE
MATTHIAS ANDREAS KAMBER, ESQUIRE
EUGENE MORRIS PAIGE, ESQUIRE
CHRISTA MARTINE ANDERSON, ESQUIRE
MICHAEL S. KWUN, ESQUIRE

KING & SPALDING LLP
1185 Avenue of the Americas
New York, New York 10036-4003
BY: BRUCE W. BABER, ESQUIRE
SCOTT T. WEINGAERTNER, ESQUIRE

GOOGLE, INC.
1600 Amphitheatre Parkway
Mountain View, California 94043
BY: RENNY HWANG, LITIGATION COUNSEL

Also Present:

SAFRA CATZ, President and CFO
Oracle Corporate Representative

CATHERINE LACAVERA
Google Corporate Representative

PROCEEDINGS

MAY 11, 2012 **7:31 A.M.**

(The following proceedings were held in open court, outside the presence of the jury.)

THE COURT: Welcome, please be seated.

How can I help you this morning?

MR. JACOBS: A couple things to report, your Honor. We are working through with Google our count of interfaces and methods and classes so that we can, hopefully, reach agreement on those numbers for your Honor.

THE COURT: When is that going to happen?

MR. JACOBS: Well, we sent over our tally late last night after going through it and describing how we had arrived at it. So, I --

THE COURT: I have been working around that piece of the order. I could put it in there right now and I could get that order out quickly possibly, so -- maybe not. It might be several weeks, but...

(Laughter.)

MR. BABER: You will have it today, your Honor.

THE COURT: All right. Fine.

MR. JACOBS: Secondly, with respect to Phase 3 issues. Oracle is not electing statutory damages on its copyright claims.

THE COURT: By the way, I don't know if it got filed today, but I am granting the Rule 50 in favor of Oracle on the seven decompiled files. That part, I think, under the instructions I gave there is only one possible answer, and that would be yes. That is the only part of the Rule 50, though, that will be granted.

Now, that having been said -- well, you don't have a damage study that ties into that infringement.

MR. JACOBS: And so we're on that legal issue, your Honor, of infringer's profits and you gave an oral ruling.

THE COURT: I'm assuming -- let me just make sure. Does Google -- formally the way this should work is we will start the trial. You can put in whatever proof you have. You cannot change your expert reports. Your expert reports don't even come close to addressing damages for the decompiled files and for rangeCheck. There will then be Rule 50 granted at the end of all of that and you won't even get statutory damages, I guess.

I don't know -- I assume that's the way it will play out, but you don't have the right to just make -- it would be the height of ridiculousness to say that for those nine lines you get hundreds of millions of dollars in profits. The law can't operate in that way.

So to my mind you're making a mistake, but that's up to you.

MR. JACOBS: Just so that we're clear on this, I think we understood that you had ruled orally in response to our filing that we were not permitted to put on that proof --

THE COURT: I'm going to let you put it on. I will take that all back. We will keep the jury here. You can put on your damages case, such as it is and ask the jury. Then I'll probably have to explain to the jury that you've got no evidence or grant a Rule 50.

If you want to go through that exercise, that's what we'll do. I was talking out loud yesterday whenever I said that you had no -- you have no proof. I don't think you have any proof that's tied in -- you've got to tie it into the exact thing that they found infringement on, which is nine lines of code. And if you can prove real damages from that, you're a great lawyer. You probably can do it if anyone can do it, but I don't think you've got a study that's tied into that.

So we will convene the trial and let you have that shot at it and at the end of that if you don't do it, we'll grant a Rule 50. We will do it procedurally correct.

Otherwise, I know what will happen. You'll go up on appeal and say that I arbitrarily and capriciously took that away from you without even looking at your evidence, when I'm not going to -- there is no need to do that. We will let you pursue the drill of trying to struggle in front of the jury. And if you think that's going to help your willfulness case in

Phase 3, that's a tactical call you've got to make, but I'm not going to take that issue away from you at this stage.

MR. JACOBS: Understood, your Honor.

May I confer with Mr. Van Nest for just a minute?

THE COURT: Sure you can.

(Discussion held off the record amongst counsel.)

MR. JACOBS: That's what I have to report to you right now on Phase 3, your Honor.

THE COURT: All right. Wonderful.

Now, both of you have been putting statements about what you would and would not waive with respect to the jury and until you come up with a written stipulation, I'm -- I don't want to get involved in what your proposals are.

I appreciate the idea that you're trying to find ways to streamline the case, but right now we're going to have a jury on Phase 3 for all issues until I see something in writing to the contrary.

MR. JACOBS: Understood, your Honor.

THE COURT: Signed by both sides.

MR. JACOBS: I have Trial Exhibit 1131, which is Professor Mitchell's demonstratives.

(Whereupon, document was tendered to the Court.)

MR. JACOBS: We would like direction from your Honor on trial testimony designations. Google intends to read from Phase 1 testimony from Mr. Kurian.

THE COURT: All right.

MR. JACOBS: And we designated completeness counters and they disagree with those being read.

So the question is: Shall we have those read all together so that Mr. Kurian's read-back from Phase 1 comes in all at once?

THE COURT: Why is that being read back at all?

MR. JACOBS: I think the Court -- well, I'll let them answer that.

THE COURT: Did I say you could do that? I thought you could do it in your closing argument, but did I say you could just repeat testimony?

MR. BABER: Yes, your Honor, you did. We had some colloquy about this last week.

THE COURT: What did I say?

MR. BABER: It's in the transcript, 2643. You said:

"You mean, just read back testimony they have already heard?"

You said:

"I would say if it's necessary to set up something, like to set up, you know, a one-two punch, you read a paragraph from some testimony if you have something that builds on that and you're reminding the jury what that prior testimony was and it's a short passage, I guess that's okay."

THE COURT: All right. I stand by that. Is that what you plan to do?

MR. BABER: Yes, sir, with Mr. Kurian it is. We have, it's basically just one page of testimony.

THE COURT: What does it set up?

MR. BABER: It sets up the testimony of Professor August with respect to differences between the Dalvik and --

THE COURT: That's fine. You can do that.

MR. BABER: And they want to designate about another page-and-a-half from before that about a meeting that happened between Google and Oracle.

THE COURT: Is it germane?

MR. BABER: We don't think so, your Honor. We think it simply puts back out there what Oracle was saying at a meeting; whereas what we're talking about is the --

THE COURT: Can I see the entire, everything that both sides want to put before the jury?

(Whereupon, document was tendered to the Court.)

MR. BABER: That's what we designated.

THE COURT: Okay.

MR. JACOBS: We have it in one document, your Honor.

(Brief pause.)

THE COURT: Mr. Kurian works for who?

MR. BABER: For Oracle, your Honor. He is an vice-president or something. He's very senior.

THE COURT: He works for who?

MR. BABER: For Oracle, your Honor. He testified right after Mr. Ellison in the copyright phase.

(Brief pause.)

THE COURT: So you want to start off by him testifying to what happened in some meeting, right?

MR. BABER: No, your Honor. Our testimony is --

THE COURT: That's what's right here. That's what you gave me. Second question:

"QUESTION: And you were asking in these meetings" --

MR. BABER: "In these meetings," that's correct, your Honor.

THE COURT: So is it the same meeting that they want to have testimony about?

MR. BABER: It is. And the testimony they want is from -- it's about a page-and-a-half from several pages earlier.

THE COURT: I think if it's the same meeting, it's close enough. We'll let them do that.

MR. VAN NEST: We'll withdraw it, your Honor.

THE COURT: So you're withdrawing 400? You're withdrawing --

MR. VAN NEST: Yes. We will withdraw it.

THE COURT: All right. So that's now moot.

Okay. What's the next issue?

MR. JACOBS: I don't think we have anything else.

MR. VAN NEST: Your Honor, a couple things. We've trimmed our witness list down. We're trying to finish this morning.

So the one thing that we had given your Honor yesterday was the video transcript designations for Mr. Gupta.

THE COURT: I returned that.

MR. VAN NEST: You did?

THE COURT: I gave it to Dawn to give to you.

MR. VAN NEST: Okay.

THE COURT: Dawn, did you give it to them?

THE CLERK: I gave it to someone at the table.

MR. VAN NEST: I'm sorry. I apologize. So we'll play that this morning.

The parties have agreed. I moved to strike a short portion of Dr. Mitchell's testimony. He testified about a HTC Droid Incredible which is no longer in the case and, therefore, the parties have agreed that the words "HTC Droid Incredible," which appear in the record transcript at Page 3463, Line 19 should be stricken.

THE COURT: That's agreed to?

MR. JACOBS: Not quite. The Droid Incredible is in the case. He testified about the Incredible 2 and -- and the testimony about the Incredible 2, because it was not linked to the Incredible, we agree that fragment should be stricken.

MR. VAN NEST: I believe the transcript says "HTC Droid Incredible." That's what I'm --

MR. JACOBS: We better -- we are trying to work this out, your Honor. I think we are getting lost in the terminology a little bit. We'll work on this again.

MR. VAN NEST: Fair enough.

THE COURT: I would prefer it if you worked it out. It sounds like it's something you can agree on.

I want to emphasize that there were things I said were out of the case because of disclosure problems earlier and I'm going to stand by that. So I don't want either side trying to slip something in.

I don't understand this problem right now well enough. If I have to get into it, I will get into it, but you all know much better than me.

MR. VAN NEST: I don't think we will need your attention on it, your Honor.

MR. JACOBS: We won't need your attention on it, your Honor.

THE COURT: All right. Good.

What's next?

MR. VAN NEST: I think we're all set to go.

THE COURT: Where were we on the witnesses? Didn't we have a witness on the stand?

MR. VAN NEST: We did. We had Mr. McFadden on, and he will be on on direct with Mr. Kamber for a portion this morning.

THE COURT: Mr. McFadden?

MR. VAN NEST: Right.

THE COURT: Okay.

MR. VAN NEST: And then following Mr. McFadden we will play the Gupta video, and following that we will call Mr. Terrence Parr, one of our experts on the '520, and and following that we will call Dr. August.

THE COURT: How come his report is so thin?

MR. VAN NEST: Not much to talk about. Not much to talk about.

THE COURT: Your reports about are about -- tiny.

MR. VAN NEST: Well, that's the '520, your Honor.

THE COURT: I'm just teasing.

MR. VAN NEST: I think we're all set.

THE COURT: All right. Let's bring in the jury and resume.

(Jury enters courtroom at 7:46 a.m.)

THE COURT: Welcome back. Please be seated.

So while you're getting your notepads out, you will remember that Mr. McFadden is the witness on the stand. We have reached the milestone where we're now in the defense case. The defense counsel say there is a good chance we're going to finish their case today, maybe all the evidence today. I'm not sure of that. Possibly we'll be going into Monday.

But I do need to give you a heads-up because of my own schedule, I have to break at about 12:15 today. So we will have a somewhat shorter day today. So that may complicate the schedule as well, but we have to break at 12:15 today. So there we are.

Are you all set over there with your notepads and your pens poised? Ready to do a work of authorship?

The floor is yours, counsel.

MR. KAMBER: I can promise less source code this morning, so...

ANDY MCFADDEN,

called as a witness for the Defendant herein, having been previously sworn, resumed the stand and testified further as follows:

THE WITNESS: I do.

DIRECT EXAMINATION RESUMED

BY MR. KAMBER:

Q. Good morning, Mr. McFadden.

A. Good morning.

Q. Welcome back.

Yesterday we walked through a demonstrative exhibit related to Resolve.c. Do you remember that?

A. Yes.

Q. And I want to change gears and talk about another functionality for the resolution of symbolic references in Android, and that is dexopt.

Are you familiar with dexopt?

A. I am.

Q. How are you familiar with dexopt?

A. I wrote it.

Q. What is dexopt short for?

A. Dex optimization.

Q. And the dex is the file format for Dalvik, correct?

A. Yes.

Q. At a high level can you please explain to the jury what dexopt does to dex files?

A. Well, it generally prepares a dex file for execution. This requires extracting the dex file from the application package that was downloaded from, say, the Android market, attempting to verify that all of the code in it is sound, and then performing a set of static optimizations.

Q. Now, approximately, when was dexopt, that functionality added to Android?

A. It sort of evolved rather than just appeared all at once.

Somewhere between middle of 2006, early 2007, I think, is when most of it appeared.

Q. I want to have you look at a document that's still on the stand with you, Mr. McFadden. It's TX 735. I believe we looked at this a little bit yesterday.

(Document displayed)

Q. Again, what does which document describe?

A. This describes the Dalvik bytecode.

Q. There is a bullet point two up from the bottom. Take a look at that, please.

And here it says:

"When installed on a running system, some instructions may be altered, changing their format, as an install-time linking optimization."

Do you see that?

A. Yes.

Q. What is this referring to?

A. This is referring to one of the things that dexopt does.

Q. Why is it referring to dexopt as an install-time static linking optimization?

A. Well, dexopt often runs at install time, and it's a static optimization.

Q. Does dexopt ever run at runtime?

A. No.

Q. Is dexopt a dynamic optimization?

A. No.

Q. We were also looking at some of the bytecode, the specific bytecodes in this document yesterday before we did the Resolve.c example, and I would like you to turn to Page 6, please.

(Witness complied.)

Q. In particular, we were looking at this box that had 52 -- that had the bytecodes 52 through, I believe, it's 5f and it says 22c. Do you see that?

A. Yes.

Q. And these are examples of instance op bytecodes, is that correct?

A. Yes.

Q. Now, we talked about -- well, can you please remind the jury what does field@CCCC stand for?

A. That's referring to an operand in the instruction. It is a -- an index value that identifies a location in the Fields table.

Q. Now, there is a note on the right-hand side here, and it says:

"Note, these OpCodes are reasonable candidates for static linking, altering the field argument to be a more direct offset."

Do you see that?

A. I do.

Q. What is that referring to?

A. That refers to replacing the CCCC value with a different value which allows you to kind of short circuit one of the steps that you do when trying to access a field.

Q. And what functionality in Android let's you do that short circuiting work?

A. Don't understand.

Q. What program does this static linking process?

A. Dexopt.

Q. Now, let's take a look at another document. This is TX 739. I believe it's on the stand with you as well. I believe we looked at this when you were called in Oracle's case as well.

Do you remember that?

A. I think so.

(Document displayed)

Q. Just to remind the jury, who wrote this document?

A. I did.

Q. Is this document part of the specification or the Android program files that OEMs use?

I'll withdraw that question. Where can this document -- this documentation be found?

A. It lives in the Android source tree.

Q. Now, about halfway down this page, based on what the jury is looking at, there is a bullet says:

"Bytecode optimization (quickened instructions, method pruning) is important for speed and battery life."

Now, "bytecode optimization and quickened instructions, "what is that referring to?

A. Now, that is referring to the thing we just looked at where the field@CCCC entry is being replaced.

Q. And what is "quickening"?

A. It refers to replacing one OpCode with a different OpCode that is intended to perform the same operation but takes slightly different operands.

Q. Let's take a look at the very bottom of Page 3 and going to the top of Page 4, Mr. McFadden.

A. Okay.

(Document displayed)

Q. The jury has already seen some of this. This is a section in this document, TX 739, regarding optimization, correct?

A. Yes.

Q. And there is a sentence at the end of the first paragraph that says:

"Some of these require information only available at runtime."

Again, what does that refer to?

A. Well, this -- this whole paragraph was talking in general terms about optimization. So that's referring to things that dexopt itself doesn't do.

Q. The sentence continues:

"Others can be inferred statically when certain assumptions are made."

What does that referring to?

A. That's referring to the things that dexopt can do.

Q. It continues:

"The Dalvik optimizer does the following."

Does the "Dalvik optimizer" here refer to dexopt?

A. Yes.

Q. And then there are two bullet points at the top for virtual method called:

"Replace the method index with a vtable index."

Do you see that?

A. Yes.

Q. What does that refer to, replacing one index with another index?

A. How deeply should I explain that?

Q. As not deeply as possible.

(Laughter.)

A. It's very similar to the field operations that we were just looking at. In this case it's a similar operation that's performed on method invocation instructions.

Q. This next bullet refers to:

"For instance field get/put, replace the field index with a byte offset."

Do you see that?

A. Yes.

Q. And "get/put," were those the instructions those instance operations -- or instance OpCodes that we were looking at in TX 735?

A. Yes.

Q. So what is this talking about:

"For instance field get/put, replace the field index with a byte offset."

A. Well, this is referring to, again, that field@CCCC operand, replacing that value with just a byte offset into an object.

Q. Again, what is the difference between an index and an offset?

A. Almost nothing. In this case the index is into a table of structured data.

When I say "offset," I'm usually referring to a byte offset, but it's still just -- you start at the top and you index or offset your way down. It's taking you straight to a location.

Q. Now, have you helped prepare a demonstrative regarding how dexopt works as well?

A. Yes.

Q. Is it shorter?

A. Much.

Q. Let's take a look at that.

(Document displayed)

Q. And here we do have a little bit of code. This is static linking using dexopt, and can you just explain for the jury what this is? And perhaps we can start with line 1953, where it says, "rewrite an IGET/IPUT instructions. These all have the form vA, vB, field@CCCC"?

A. So what this is saying is just the purpose of this function, which is to take one of these IGET or IPUT instructions and rewrite it.

The note about the form is just a reminder to myself of how the instruction is laid out in memory.

Q. Now, there's a highlighted line here, 1963. What is this line doing where it refers to "rewrite inst field"?

A. Well, that's the name of the method and the arguments it takes.

Q. It goes on to Line 1966. This is, "u2 fieldIdx = insns [1]."

Can you decipher that for the jury please?

A. This is pulling the field index out of the instruction stream. So this is getting that field@CCCC value and putting it into a variable called fieldIdx.

Q. What happens at line 1970?

A. This is a call to dvmOptoptResolveInstField, which is very similar to dvmResolveInstField, but because this is part of the optimization process, it has to work a little bit differently. So the "opt" in the name indicates it's the variant that's used for optimization.

Q. Now, jumping down a little on the page to Lines 1983 and 1984, what's happening here?

A. So this is where the instruction rewriting actually takes place. Line 1983 is replacing the OpCode. So, for example, IGET would be replaced with IGET-quick.

Line 1984 is where the field index is being replaced with a byte offset.

Q. It says something at the very end of line 1983. It says, "new Opc." What does that stand for. '?

A. New OpCode.

Q. So let's go back to the demonstration that we saw yesterday and sort of where we left off when we talked about resolve.c to start talking about dexopt.

Again, what is on the left-hand side?

A. That is the instructions stream.

Q. And what is on the right-hand side?

A. A whole bunch of data.

Q. Now, there is a pointer, this -- well, let me ask: What is this number 21 in the Resolved Fields table at entry one?

A. That is a pointer to a field structure.

Q. And how does dexopt use the pointer to the field structure?

A. Well, it's going to follow that to get at the information about the field.

Q. What information about the field does dexopt use?

A. It utilize the byte offset, which in this case is now showing up. Yeah, the very bottom entry.

Q. The byte offset is 48 on this graph?

A. Yes.

Q. At the bottom of that Field object, is that correct?

A. Yes.

Q. Now, what happens in the process now of dexopt and this conversion from an IGET process -- IGET instruction to an IGET-quick instruction?

A. Well, at this point we're going to replace the OpCode and operand with the new values.

Q. So let me do that again just so that the jury sees.

There is a 52 at the top left, correct, in the instruction stream?

A. Yes.

Q. And that gets rewritten as a 242, correct?

A. Yes.

Q. What is happening there?

THE COURT: Wait, wait. I didn't understand how that happened. How did the 52 go to 242? Explain that.

THE WITNESS: From the very bottom of the code sample we were just look at where it is taking New Opc and using that to replace the OpCode, changing it from IGET to IGET-quick.

THE COURT: I don't see 242 anywhere else on the whole page. I don't see where it even came from.

THE WITNESS: It is not on this page.

BY MR. KAMBER:

Q. Mr. McFadden, where does the number 242 come from? What does it represent?

A. Well, it is the OpCode number.

MR. KAMBER: Your Honor, I'll try to explain this.

BY MR. KAMBER:

Q. There was an OpCode there that was named 52 before, correct?

A. Yes.

Q. And what does that OpCode -- what instruction does that OpCode tell the computer to perform?

A. 52 is IGET.

Q. Now, is there is OpCode called IGET-quick?

A. Yes.

Q. What is the number for IGET-quick?

A. Hopefully, 242.

Q. And so what is happening in this process of rewriting the bytecode from 52 to -- excuse me, from rewriting the OpCode from the number 52 to the number 242?

A. Well, the dx optimizer knows the values of all of the OpCodes. So as it's doing the rewriting, it knows that it's going to take this 242 value and drop it in where the 52 was before.

Q. Now, there's the 48 we were looking at at the bottom of the Field table. That's an offset, correct, in the bottom of the Field table?

A. Yes.

Q. And what happens with that 48?

A. That replaces the operand 01.

Q. Let me do that again so that everyone can see it again.

So the offset is going from that field and -- is that being used to rewrite the operand in the instruction in the process of dexopt?

A. Yes.

Q. TX 739 refers to dexopt as a back door into the VM. I believe it's on Page 3.

(Document displayed)

Q. It's in that second paragraph. It says:

> "The solution is to invoke a program called
> dexopt, which is really just a back door into
> the VM."

What were you trying to explain here, Mr. McFadden?

A. Well, the -- most of the Dalvik Virtual Machine lives in a library. It's actually largely built as a shared library.

So what I was just trying to say here is I am reusing a lot of the code that is in the VM, but if you run -- if you launch the VM and you launch dexopt, they come in from different directions.

So all I was really trying to say was if you're running the VM, you're kind of coming through the front door. If you're running dexopt, you're coming in through the back door and they just have a lot of shared code.

Q. As far as the resolution of symbolic references in the dex file goes, does dexopt -- do dexopt and Resolve.c work differently?

A. They do some very similar things.

Q. In what sense?

A. Well, the actual resolution code, while not exactly the same, is very similar.

Q. When you're talking about the resolution code, are you talking about the demonstrative that we looked at yesterday where we played connect the dots through the indexes to a symbol in a table?

A. Yes.

Q. Is that process generally the same in dexopt and Resolve.c?

A. Yes.

Q. Let's turn to TX 737.

(Document displayed)

Q. I believe we looked at this yesterday as well.

There is a statement on Page 2 towards the bottom.

MR. KAMBER: If you could highlight those two paragraphs, Ben, the one that starts "Arguments"?

(Document highlighted)

MR. KAMBER: Thank you.

BY MR. KAMBER:

Q. (As read)

> "Arguments which indicate a literal constant
> pool index have the form kind@x."

And we discussed this yesterday. This is referring to indexes into the constant pool tables, correct?

A. Yes.

Q. Now, there is a paragraph below that that starts:

> "Similar to the representation of constant
> pool indices, there are also suggested
> (optional) forms that indicate prelinked
> offsets or indices."

Do you see that?

A. Yes.

Q. What does that refer to?

A. This is referring to the format of the instructions after dexopt has done its optimizations.

Q. And it continues:

"These prelinked values include "vtaboff."

Do you see that?

A. Yes.

Q. What does "vtaboff" stand for?

A. Vtable offset.

Q. That's reflected in this document, correct?

A. Yes.

Q. And "field off," that refers to field offset?

A. Yes.

Q. And "iface," what does "iface" refer to?

A. Interface pool index.

Q. Just so it's clear, the first paragraph talks about what a -- well, the first paragraph talks about what a dex file looks like when it arrives on a phone, is that correct?

MR. JACOBS: Your Honor, I think the -- a lot of leading going on.

THE COURT: It is leading. Please rephrase the question.

BY MR. KAMBER:

Q. What does the first paragraph have to do with -- how does it relate to dex files when they arrive on a device?

A. I'm not quite sure how to work with "arrive on a device."

What I can say is that it's describing how the code looks when it is generated by dx.

Q. And what does the -- this paragraph that starts "Similar to the representation" refer to? What form of the dex file?

A. Oh, well, once you've got the instructions replaced, then that is what we usually refer to as an .odex file for optimize dex.

Q. Does the second paragraph -- is the second paragraph talking about the optimize dex file after it has been processed by dexopt?

A. Yes.

Q. Let's turn to the third page.

(Document published to the jury.)

Q. We were looking yesterday, there is a section "The Formats" relating to bytecode forms, and we were looking at an OpCode about two-thirds of the way down the page, an OpCode format. 22c, this is "instance-of."

And, again, for the jury, what are we look at here with respect to this line 22c?

A. That's describing the instruction format, which, among other things, is used for the IGET/IPUT instructions.

Q. Directly below that it has an instruction format that says, "22cs op vA, vB, field off@CCCC." What is that referring to?

A. This is referring to the format of the instructions after they have been quickened by dexopt.

Q. On the right-hand side there is a note in parentheses:

"Suggested format for statically linked field access instructions of format 22c."

Do you see that?

A. Yes.

Q. What is that referring to?

A. Well, this is just saying that for -- for the types of optimizations that dexopt does, this is the recommended format.

Q. Is field@CCCC the index that's in the -- the operand in the bytecode instruction when it arrives on the dex file?

MR. JACOBS: Your Honor, leading.

MR. KAMBER: It's --

THE COURT: It is leading. Sustained.

"To what extent, if at all...."

MR. KAMBER: Excuse me?

THE COURT: "To what extent, if at all..."

MR. KAMBER: Thank you. Thank you, your Honor.

BY MR. KAMBER:

Q. To what extent, if at all, does the field@CCCC refer to the operand in the instruction stream when it arrives on the device -- or in a dex file? Better said.

A. I have no idea what that means. I'm sorry.

Q. That's not a problem, Mr. McFadden.

I'm just -- I'm just trying to get at what this change is that's happening through dexopt. Can you explain whether and how dexopt changes field@CCCC to fieldoff@CCCC?

A. Okay. That was the code that we were just looking at.

Whereas, part of this quickening operation it is replacing the field index value with a field offset value in the instruction stream.

Q. To what extent, if at all, is field@CCCC a location in memory in the dex file?

A. I'm struggling a bit with "location in memory in the dex file." We are rewriting the file, so saying it's in memory is a little strange.

What I will say is it does identify a location in a Field table.

Q. Does fieldoff@CCCC identify a location it?

A. It does.

Q. How does it do that?

A. It's an offset into an object.

MR. KAMBER: No further questions, your Honor.

BY MR. KAMBER:

Q. Thank you, Mr. McFadden.

CROSS EXAMINATION

BY MR. JACOBS:

Q. Good morning, Mr. McFadden.

A. Good morning.

Q. Do you have 735 in front of you?

A. Yes.

(Document displayed)

Q. If you go down to the first page towards the bottom, there is a passage that you discussed a few minutes ago:

> "When installed on a running system, some instructions may be altered, changing their format as an install-time static linking optimization."

Do you see that?

A. I do.

Q. And when it says "some instructions may be altered" that's a reference to dexopt?

A. Yes.

Q. The instruction alteration referred to in 735 occurs during installation on a running system?

A. Not necessarily.

Q. Well, this passage reads:

> "When installed on a running system, some instructions may be altered."

Is that passage accurate?

A. It is -- that is a true statement, but it is incomplete.

Q. In what way is it incomplete?

A. Well, the instructions may be altered as the system is being built.

Q. That's on the developer's side?

A. Yes.

Q. So when we're talking about dexopt running on an actual phone, then we're talking about this sentence; that is,

"...installed on a running system;" correct, sir?

A. Yes.

Q. And that was the focus of your -- well, that's dexopt running on a phone. That's a running system?

A. Yes.

Q. And so this instruction alteration referred to in TX 735 referred to in this sentence, that's occurring during installation on a running system?

A. Yes.

Q. And there is a sentence:

"This is to allow for faster execution once linkage is known."

Do you see that? Right after the sentence we were talking about.

A. Yes.

Q. And so dexopt's instruction rewriting allows for faster execution?

A. It can.

Q. This sentence is true? This is to allow for faster execution once linkage is known; true, sir?

A. Yes.

Q. Dexopt's instruction rewriting can only be performed on a device, on a handset once the linkage is known; true?

A. True.

Q. It's also true that the linkage can only be known --

again, on the device side -- on a running system?

A. If by "running system" you mean it's plugged in and turned on, then I agree with you.

Q. I mean, "running system" as expressed in the sentence.

A. Well, if it's plugged in and turned on, then it's a running system. So, yes.

Q. I'd like to look at Slide 20 of your slides from yesterday.

(Document displayed)

Q. Now, you testified that what's shown here is an simplified depiction of a dex file. Do you recall that testimony to the jury?

A. This is more than just a dex file because it includes the Resolved Fields table.

Q. So the portion to the right of the Resolved Fields table and the instructions, that is, by your testimony, a simplified depiction of a dex file; true, sir?

A. True.

Q. For the Resolved Field table, that's not part of a dex file; true, sir?

A. True.

Q. But the portions underneath "Data," identified as data on this slide, they are part of the dex file; true, sir?

A. Yes.

Q. And just to explain again, the dex file is what's created by the operation on a source code file of the Java compiler and the dx tool; true, sir?

A. I don't quite understand your question.

Q. The dex file that we're talking about here is the bytecode version of the application programmer's program; correct, sir?

A. Yes.

Q. And that got created because somebody wrote a program as an application programmer, ran it through the Java compiler, ran it through the Android dx tool, and then in the case of dexopt running on a handset, that bytecode got installed -- got delivered to the handset?

A. Sounds about right.

Q. That's how we get to this place of having a dex file that we're looking at the form of; true, sir?

A. Yes.

Q. Now, let's look at 736 again. This is your document .dex. The Dalvik executable format?

A. I would point out this is not actually my document.

Q. But you're familiar with it?

A. I am.

Q. And you stand by it?

A. Sure.

Q. So if your slides represent a simplified depiction of a dex file, TX 736, that's a simplified description of the dex file format; true?

A. Yes.

Q. And, in fact, that was your testimony; that this document describes the format of dex files, and you stand by that testimony; true, sir?

A. Yes.

Q. In fact, at the very top of the document it says:

"This document describes the layout and contents of .dex files."

Do you see that?

A. Yes.

Q. And that's a true statement?

A. Yes.

Q. Let's turn to the second page of 736 and the section entitled "Overall File Layout."

(Document displayed)

Q. Do you see that, sir?

A. Yes.

Q. And this is a depiction of the layout of the contents of a dex file; true, sir?

A. Yes.

Q. This shows that a dex file actually has nine sections; true.

A. Yes.

Q. And the first section is called "Header"?

A. Yes.

Q. And then there are five ID sections: String_ids, type_ids, proto_ids, field_ids and method_ids.

Do you see that?

A. Yes.

Q. The field_id section of the dex file, that's the Field ID table that you have testified about; true, sir?

A. I believe so.

Q. In fact, if we go back to your slide.

(Document displayed)

Q. The Field ID table listed under Data on your slide is the second column in the green area of your slide; true, sir?

A. I'm just double checking the field_id item definition.

(Brief pause.)

A. Yes.

Q. And so you show it under Data on your slide, Field ID. That's the Field ID section that's referred to as the fifth section of the overall file layout on the .dex Dalvik executable format, Trial Exhibit 736; do you see that, sir?

A. Yes.

Q. And the other sections you talked about, likewise, correspond to tables you have testified about; true, sir?

A. I believe so.

Q. So the string_id section in the dex file format is the String ID table you testified about earlier, true?

A. Yes.

Q. Now, following the five ids sections, there's a class_defs section; true, sir?

A. Yes.

Q. And following the class_defs section in a dex file is what the overall file layout portion of 736 refers to as the Data section; true, sir?

A. True.

Q. And the description of the Data section is:

"Data area containing all the support data for the tables listed above."

And that's an accurate description; yes, sir?

A. It is.

Q. So what this description of the overall file layout of a dex file shows is that the Field ID table is not stored in the Data area of a dex file; true, sir?

A. It's not stored in the section that's labeled "Data."

Q. Not stored in the section labeled "Data" by TX 736, Google's official definition of the dex file format; true, sir?

A. True.

Q. Now, if we go back to 20, Slide 20.

(Document displayed)

Q. You mentioned this earlier, but just to make sure we cover it. The Resolved Fields table is not part of a dex file also; true, sir?

A. True.

Q. So you didn't mean to convey that the Resolved Fields table is part of some kind of data area of a dex file when you showed this to the jury; correct, sir?

A. Correct.

Q. Now, staying with Slide 20 for a minute.

OpCode 52, that's the IGET instruction?

A. Yes.

Q. Field Index 01, that's an index to the Field ID table?

A. Yes.

Q. And what this connect-the-dots chart shows is that Field Index 01 corresponds to a field named "fun" of type "byte," correct?

A. Eventually, yes.

Q. And the field is a variable and its name is "fun."

The field at 08 under String Data is a variable and its name is "fun"?

A. I wouldn't refer to it as a variable. I would just refer to it as a field.

Q. Well, you used the word "constant" yesterday and I think that got a little confusing. It's in the constant pool table; true, sir?

A. It is in the dex file's constant pool.

Q. And the reason it's a constant is that "fun" remains "fun," not "funnily," not "funny." It's "fun." It's always "fun;" true, sir?

A. True.

Q. But the actual value, the actual field value, your language yesterday, could vary depending on the operation of the program?

A. True.

Q. Now the way "fun" gets in this dex file is that a programmer writes a program that contains a statement with "fun" in it; true, sir?

A. True.

Q. Something like "y = fun + 2."

THE COURT: "Fun" plus -- is "fun" supposed to be a number or a string? Can we go back over that part again?

BY MR. JACOBS:

Q. "Fun" is a string; true, sir?

A. True.

Q. And it represents -- it stands for ultimately an actual field value; true, sir?

A. No.

Q. How would you say it?

A. "Fun" is the name of a field. So if you have a class, classes have methods and fields. One of the fields is named "fun."

THE COURT: Would that have a numerical value like 10 or would it have a string value like, you know, your name?

THE WITNESS: It just depends on how the class is declared. Every field has a type and a name. So if I say "fun" is an integer field, then it can hold a number. If I say "fun" is a string field, then the field would hold a string.

THE COURT: You would then have to have a dollar mark at the end to be a -- it could look exactly that way? The name would be exactly the same?

THE WITNESS: Yes.

THE COURT: All right.

THE WITNESS: The name and type are distinct in the Java programming language.

THE COURT: Okay. I'm sorry for the interruption, but let's have that point clear and then continue on.

Go ahead.

BY MR. JACOBS:

Q. In the case at hand "fun" represents a numerical -- ultimately will represent an integer value; true, sir?

Let me back up. You referred yesterday to an actual field value; do you recall that?

A. Not specifically, but I'll -- I'm perfectly happy to go with that.

Q. That's a vocabulary that makes sense to you in this context; correct sir?

A. Sure.

Q. And "fun" has an actual field value?

A. "Fun" is just the name of the field. It doesn't have a value.

Q. The field that "fun" refers to has a value?

A. Yes.

Q. Ahh, okay. Now we're in sync. And "fun" is a symbol representing a field that has a value?

A. "Fun" is just a symbol. I could have a method called "fun." I could have 10 -- 10 different classes with fields called "fun." And they would all use the same symbol because in a dex file all instances of the word "fun" are shared.

Q. In this particular case, the illustration you gave to the jury, "fun" is a symbol for a field that has a value?

A. Yes.

Q. And what's the value of that field in your illustration?

A. The value of the field is not shown in the illustration.

Q. But it was shown; correct, sir?

A. I don't believe so.

Q. Again, the way we get to "fun" is that the programmer writing a source code program decided to name a -- to give a name "fun"; correct, sir?

A. To name a field "fun"? Yes.

Q. Okay.

A. Yes.

Q. Okay. So the programmer decides to name a field "fun." It goes through the Java compiler. It go through the dx tool. It gets into a dex file, and it's still "fun." True, sir?

A. Yes.

Q. That's how we got it. The programmer chose that name "fun"?

A. Yes.

Q. And he could have chosen another name, it's just we're having fun with "fun," so "fun" is our variable; true, sir?

A. Yes.

Q. Some of us are having fun with "fun."

(Laughter)

Q. Let's go to 647 -- sorry. Let's go to Trial Exhibit 46.6. Actually, let me -- I think I can shortcut this.

Let's say you had an object that has a "fun" field, and it stores a value of 17. Okay?

A. Okay.

Q. Then there's going to be -- go back to slide 20. And we already discussed that 52 is the IGET instruction, right?

A. Yes.

Q. And the result of the IGET instruction would be to obtain the actual value of 17, in that case; true, sir?

A. True.

Q. So the IGET instruction, the role of the IGET instruction is to obtain actual field data from an object; true, sir?

A. True.

Q. And what the IGET instruction does is obtain actual field data from an object and store it in a Dalvik register?

A. Yes.

Q. So that IGET stores the actual field data into the register; true, sir?

A. Yes.

Q. And so if we -- looking at slide 20, we have an IGET instruction with "01" as the field index. The IGET instruction doesn't obtain the number "01" and store "01" in a Dalvik register; true, sir?

A. True.

Q. The IGET instruction doesn't obtain the number "2," shown here under field ID, or "76" from the string ID table, and store "2" or "76" in a Dalvik register; does it?

A. It does not.

Q. The IGET instruction doesn't obtain the name "fun" and store that in a Dalvik VM register; does it?

A. True.

Q. It doesn't obtain the name "byte" and store that in a Dalvik Virtual Machine register; does it?

A. No.

Q. The actual data that is obtained by the IGET instruction and stored is the value of the field named "fun" in an object; true?

A. True.

Q. That value, the actual field data in an object -- this is the point you were making before -- that's not even shown here on slide 20; is it?

A. No.

Q. In my example, where the actual value of the field in an object was 17, the number 17 would not appear on this slide the way you drew it?

A. Correct.

Q. The field index in the IGET instruction is the -- not the numerical memory location of the actual field data in an object; is it, sir?

A. It is not.

Q. So, now, let's look at the IGET instruction in 735. Take a look at page 6.

A. Okay.

Q. True or false, the Dalvik IGET instruction never contains the numerical memory location of the actual field data that it is supposed to get, and ultimately store it in a Dalvik register?

A. True.

Q. The IGET instruction obtains an opcode followed by three operands. The operands are VA, VB and field@CCCC; true?

A. True.

Q. True or false, the VA operand is not the numeric memory location of the actual field data, the actual value of the field?

A. True.

Q. The VB operand is not the numeric memory location of the actual field data, the actual value of the field?

A. True.

Q. The field@CCCC is not the numeric memory location of the actual field data, the actual value of the field?

A. True.

MR. JACOBS: Thank you, Mr. McFadden.

THE COURT: All right. Any redirect?

MR. KAMBER: Briefly, Your Honor.

REDIRECT EXAMINATION

BY MR. KAMBER:

Q. Mr. McFadden, let's go back to Trial Exhibit 735, where Mr. Jacobs started off, and that bullet point at the bottom that starts "When installed on a running system." I've got some questioning there.

What do you mean -- what do you mean -- what does this mean by "running system"?

A. Uhm, it's the -- the basic definition of running. You know, power is turned on, battery's inserted.

Q. Does this mean that the system is executing Dalvik bytecode?

A. Not necessarily.

Q. It's just -- well, there's a lot of discussion about running and runtime. Is dexopt a static or a dynamic optimization?

A. It performs static optimizations.

Q. Why is it called a static optimization?

A. Because it doesn't require information that is only available at runtime.

Q. And when you use the word "runtime" in that answer, how are you using it?

A. Uhm, in the sense that -- well, there's different -- things happen at different times. So there's compile time, where the compilers in dx are running.

There's install time, when packages are being downloaded to the device and installed.

And then there's runtime when the application itself is actually executing.

Q. Does dexopt operate when the Dalvik bytecode is actually executing?

A. No.

Q. Is that why you use the term -- why you say it's not a dynamic process?

A. Yes.

MR. KAMBER: Ben, could we have slide 20 from Mr. McFadden's presentation, please. Or 21. That's fine. Thank you.

BY MR. KAMBER:

Q. This demonstrative, you had some questions about instructions and data on the data section. Did you intend for this demonstrative to map exactly the dex file format document that you also looked at, Exhibit 736?

A. Well, it's not intended to be exactly anything. It's certainly not meant to exactly mirror the contents of a dex file.

Q. Why did you use the term "data" here, when referring to the field ID table or the string ID table, or the string data?

A. To contrast it with the -- the instruction stream.

Q. Is the instruction stream separate from these tables and the data in these tables?

MR. JACOBS: Leading, again, Your Honor.

THE COURT: All right. Try not to lead.

MR. KAMBER: Sorry, Your Honor.

BY MR. KAMBER:

Q. To what extent are the instructions in these tables separated in the dex file?

A. Uhm, it's, uhm -- well, it's all kind of -- a lot of things are interleaved, but the instructions for a given method are distinct. They occur in a solid block inside the dex file. So you can, you know, find a chunk, which is the instructions for a given method.

Q. And let's take a look at, again, at that exhibit, 736. We talked about this yesterday.

On page 17, there's a section titled "insns." Do you see that?

A. I do.

Q. Is that the chunk you were just referring to?

A. Yes.

Q. Now, let's go back to that slide 21, please.

Mr. Jacobs asked you if the -- the "1" in the instruction stream corresponds to the "fun," the symbol in the string data. Do you remember that?

A. I think so.

Q. Now, what does the -- what does the operand 1 tell you?

A. Well, it's simply an index into the field IDs table.

Q. Does it tell you a location?

A. Yes.

Q. What happens when you get to the location "1" in the field ID table?

A. You read the data there, which are also index values. And you chase those to the next location.

Q. And what happens at the next location, or those next locations in the string ID table?

A. Same -- same thing.

Q. So you're following an instruction to go from one location to another location to another location?

A. Yes.

Q. What happens when you reach the string data table?

A. Well, at that point, you're no longer working with numeric values. You've got string data. And you have to take those and use them to find a matching field.

Q. Is string data, is it sometimes called a symbol?

A. Yes.

Q. Is it sometimes referred to as a symbolic reference?

A. Yes.

Q. Is that distinct from numeric references?

MR. JACOBS: Your Honor, we're leading again.

BY MR. KAMBER:

Q. How is that --

THE COURT: Please, don't lead so much. These are important points, and you should not be leading so blatantly.

MR. KAMBER: I'm just trying to ask an is question, rather than saying --

THE COURT: That's leading. In this context, it is highly leading. Thank you.

MR. KAMBER: Thank you, Your Honor.

BY MR. KAMBER:

Q. What is the difference between a -- a reference where you use a symbol like this and a reference by location like the indexes that we've been chasing through?

A. Uhm, well, the -- the numeric references, the indexes, the offsets, they take you directly to the next place you need to be.

With the symbolic references, the strings like "fun" and "byte," they don't give you an address. You actually have to take them and compare them against something else. So you need to find -- you need to find the field that matches them. And you can't just go straight there. You have to go and kind of walk through the class data and search for it.

MR. KAMBER: No further questions, Your Honor.

THE COURT: All right. May the witness -- there's more to go?

MR. JACOBS: Your Honor.

THE COURT: All right. Go ahead.

RECROSS EXAMINATION

BY MR. JACOBS:

Q. Take a look at 739, page 3.

A. Okay.

Q. Just for orientation the following statement is true? When dexopt runs, it has loaded the application it is optimizing into the Dalvik VM.

MR. KAMBER: Your Honor, I think this is beyond the scope of redirect.

MR. JACOBS: It's about dynamic, Your Honor.

THE COURT: Was that asked about just now?

MR. JACOBS: Yes, he asked him specifically what dynamic was.

THE COURT: All right. Overruled. Please, go ahead.

BY MR. JACOBS:

Q. When dexopt runs, it has loaded the application it is optimizing into the Dalvik VM?

A. For a limited definition of Dalvik VM, yes.

Q. And when dexopt runs, it depends on information -- when dexopt runs on the handset, it depends on information about the configuration of the handset; true, sir?

A. True.

Q. And, in fact, when updates are sent to the handset you have to rerun dexopt?

A. True.

Q. So that's like when -- a courthouse, in which after a trial is over the courtrooms reconfigure, and Courtroom 8 moves over to Courtroom 9, and you have to check the actual conditions of the courthouse to figure out where courtroom 8 is; isn't it, sir? Maybe you weren't here for my dramatic reading of my analogy.

THE COURT: I don't think we should --

MR. JACOBS: I withdraw that question, Your Honor.

THE COURT: That was hard to follow.

MR. JACOBS: That was hard to follow.

MR. VAN NEST: Would you be moving, Your Honor?

THE COURT: I don't know. They may be trying to tell me something.

(Laughter)

BY MR. JACOBS:

Q. The layout of the memory, which Dalvik needs to know, changes from time to time; true, sir?

A. I'm not quite comfortable with "the layout of the memory."

Q. The actual locations of objects that dexopt needs to find.

A. Uhm, the locations of objects are irrelevant for dexopt.

Q. The locations of actual field values changes?

A. The actual location of field values is not known to dexopt.

Q. It's not known to dexopt. But dexopt resolves what even you concede are symbolic references?

A. True.

Q. And it resolves them into numerical references?

A. True.

Q. And because that resolution process depends on the conditions actually existing on the handset, when those conditions change by way of a system update, dexopt needs to rerun?

A. True.

Q. And in that sense it's dynamic?

A. No.

Q. Because you disagree with what "dynamic" means?

A. I'm not sure what you mean by "dynamic," but it is possible that we disagree.

Q. If I mean by dynamic, depending on conditions on the handset which can change from time to time, then it is dynamic; true, sir?

A. Okay.

MR. JACOBS: No further questions, Your Honor.

MR. KAMBER: Your Honor, I just have one quick follow-up question.

FURTHER REDIRECT EXAMINATION

BY MR. KAMBER:

Q. Mr. McFadden, you were struggling to answer the question about whether the actual data or the object exists at the time of dexopt. Why did you have trouble answering that question?

A. Uhm, because while dexopt is running, the application itself is not running.

So to talk about the objects or the field contents for the application doesn't make any sense. The object -- the application can't have created any objects because it's not actually executing.

Q. Do the objects exist at the time that dexopt is running?

A. No.

MR. KAMBER: No further questions.

THE COURT: All right. May the witness step down?

MR. JACOBS: Yes, Your Honor.

THE COURT: All right. Mr. McFadden, thank you.

MR. JACOBS: Your Honor, Mr. McFadden is subject to recall.

THE COURT: All right. You are subject to recall, so we can't let you go for good, but for now.

THE WITNESS: Thank you.

THE COURT: We'll let you know if you are needed.

(Witness steps down.)

THE COURT: Next witness, please.

MR. VAN NEST: Your Honor, I believe at this time we are going to play a short videotape deposition from testimony from Vineet Gupta, who was an employee of Sun and later of Oracle.

THE COURT: All right. So how long is it going to be, ten minutes, less?

MR. PURCELL: About eight minutes, Your Honor.

THE COURT: You all know the drill by now. Just sit back and take notes. Play it.

MR. PURCELL: Your Honor, just one thing for the jury's reference. There's a discussion of a document toward the end of the clip, which is in evidence. That's TX 2072. It's not referred to in the colloquy, but it is in evidence.

THE COURT: Thank you. All right. We'll all pay attention.

WHEREUPON:

VINEET GUPTA,

called as a witness for the Defendant herein, testified via videotaped deposition played in open court in the presence and hearing of the jury.

(Time noted: 8:50 to 8:57 a.m.)

MR. VAN NEST: That concludes the deposition, Your Honor.

THE COURT: All right. Next witness.

MR. PAIGE: Your Honor, Google calls Dr. Terence Parr to the stand.

THE COURT: All right. Is that you?

THE WITNESS: Yes, it is.

THE COURT: All right. Please raise your right hand.

TERENCE PARR,

called as a witness for the Defendant herein, having been first duly sworn, was examined and testified as follows:

THE WITNESS: I do.

THE CLERK: Okay. Thank you.

THE COURT: You need to sit about this close. See how the mic moves all around?

THE WITNESS: Yes, I do.

THE COURT: Say your name.

THE WITNESS: Terence Parr.

THE COURT: That's perfect.

You may go right ahead.

Can you go another 15 minutes before the break?

All right we'll go about 15 minutes. You find a good, convenient breaking point and then we'll take the break.

MR. PAIGE: I'll do that, Your Honor.

THE COURT: Thank you.

DIRECT EXAMINATION

BY MR. PAIGE:

Q. Good morning, Professor Parr. How are you?

A. I'm okay. I'm a little chagrin to see another hardwood chair here.

Q. Can you introduce yourself to the jury and let them know what you do for a living.

A. Sure. My name is Terence Parr. I'm a professor and graduate program director in computer science at the University of San Francisco.

Q. And can you briefly relate your educational background.

A. Sure. I have a bachelor's degree in computer science, and a Ph.D. in computer engineering from Purdue University.

Q. What did you do after you obtained your Ph.D. from Purdue University?

A. I went to work for the Army High Performance Computing Research Center at the University of Minnesota.

Q. And when was that?

A. I graduated in 1993, and so moved to Minnesota right after that.

Q. Now, the jury has already heard a great deal about Java in this case. Are you familiar with the Java Programming Language?

A. Yes. Very familiar. I've written hundreds of thousands of lines of Java. And, in fact, I've been writing Java since before the formal announcement. So about 17 years.

Q. Where do you live today Professor Parr?

A. Here in San Francisco, in the Mission.

Q. And why did you move to San Francisco?

A. Well, once I started playing around with Java, and the Internet was really starting to boom, I decided something very interesting was going to happen with the confluence of this sort of Java executable stuff with the Internet. So I moved out here quickly.

Q. And when was that?

A. That was 1995.

Q. What did you do after moving out to San Francisco?

A. Within a few months I started a company called jGuru, which was, essentially, a training company that would help big companies such as Microsoft, FedEx, things like that, retool their engineers.

So, in other words, everyone has been programming in C++, let's say, and everyone is jumping on the Java bandwagon. And so we trained engineers how to program in Java.

Q. And what was that company called?

A. That was jGuru.com. Effectively it was our doing business.

Q. Who wrote the server that ran that website?

A. Yes, as part of this we had a very large website which I wrote. It's between a hundred and 150,000 lines of Java code, depending on how you count.

Q. Now, the jury has also heard a lot about virtual machines in this part of the case. What experience, if any, do you have with virtual machines?

A. Well, in my graduate classes and undergraduate classes, I teach how to build virtual machines. I've built dozens of them. And perhaps the biggest one is something I did for Renault Automation.

(Reporter interrupts.)

A. Renault Automation. Renault. I'm not sure how to say it in English. For Renault Automation. It's the robotics arm of the French car manufacturer.

And I worked in Paris for a while, to produce a debugger, compiler and virtual machine that was fundamentally the same as the Java Virtual Machine. It's -- it was building cars for many years, as far as I know.

Q. Have you written any books related to computer programming?

A. Yes. I've written three books on language implementation and parsing. And I'm working on my fourth right now.

Q. And are you involved in any work that many Java programmers use?

A. Yes. I'm perhaps best known for a tool called ANTLR. It stands for ANother Tool for Language Recognition. It's used by essentially, all major companies. It comes on every Linux machine, and the developer kit for Apple. It's widely used.

Essentially, it's to help programmers build parsers. In other words, a parser is something that identifies the parts of speech in an English sentence, for example. So that's the verb. That's the object.

MR. PAIGE: Your Honor, may I approach?

BY MR. PAIGE:

Q. Professor Parr, I'm handing you Trial Exhibit 2672. Could you take a look at that and tell me what that is?

A. Yes. This is my curriculum vitae.

Q. What is your curriculum vitae?

A. It's a long, boring list of my education, papers, projects, things like that.

MR. PAIGE: Your Honor, we offer 2672 into evidence.

MR. JACOBS: No objection.

THE COURT: Received.

(Trial Exhibit 2672 received in evidence.)

MR. PAIGE: Your Honor, we tender, at this time, Professor Parr as an expert in the field of computer science.

MR. JACOBS: No objection.

THE COURT: All right. As long as we stay within the scope of his expertise, we'll be fine.

Please continue.

MR. PAIGE: Thank you, Your Honor. May I approach again?

THE COURT: You may.

BY MR. PAIGE:

Q. Professor Parr, I've handed you Trial Exhibit 4011, United States Patent No. 6,061,520. Are you familiar with the '520 Patent, Professor Parr?

A. Yes, I am.

Q. And is this Trial Exhibit 4011 the '520 patent that is at issue in this case?

A. Yes.

Q. Can you explain some of the background of the patent? I would like you to start by looking at the abstract of the patent --

A. Okay.

Q. -- where it talks about "statically initializing an array."

A. Yes. I see that.

Q. What is an array?

A. An array is just a list of memory locations, memory cells we can put things.

Q. And what's, generally, contained in the memory locations in an array?

A. Uhm, you can put, you know, references to strings and any sort of object you want. But often we put numbers in there.

Q. And could you explain to the jury the difference between creating an array on the one hand and initializing an array on the other?

A. Yes. To create an array means to allocate space within the memory for these cells that I can put values. To initialize means to specify the default values. So when the program starts, what should the default values be?

Q. And what does it mean to execute a program?

A. Well, it's like when you launch Microsoft Word on your laptop or, you know, you have an Android phone and you click, you know, an app to start it up. It means to follow a recipe. The programmer has creating a recipe, a program, and you follow the steps of that.

Q. Now, have you prepared a slide showing how an array is initialized using Java bytecode?

A. Yes.

THE COURT: One second. I'm sorry, 4011 is the patent. It's in evidence, right?

MR. PAIGE: I believe so, Your Honor.

THE COURT: It should be shown to the jury, yes. I'm sorry for the interruption. Go ahead.

MR. PAIGE: Could we put up slide 1, please.

(Document displayed.)

BY MR. PAIGE:

Q. Now, what does slide 1 show, Professor Parr?

A. Slide 1 shows a number of things. On the left we can see some code in the Java programming language. It defines a static array called "tester" with four initial values: 1, 2, 3, 4.

And then we have a depiction of the Java compiler, which grinds away on the source code and generates these Java stack machine instructions that you see on the right.

Q. Can you explain to the jury what the various instructions on that right-hand side mean?

A. Sure. At address 0 -- we see there "0:" -- we see something called "iconst_4." That pushes a constant value 4 onto the stack.

Q. Now, what's a stack?

A. Oh, a stack is sort of like the scratch pad for a stack machine. It performs all operations on the stack. Every instruction you see here manipulates the stack in some way.

Just to give you an idea how a stack machine operates, imagine that we want to add two numbers, say 1 and 2. A stack machine pushes a 1 onto the stack. Imagine a stack of plates. So I put a 1 plate on the stack. And then I push a 2 plate on the stack. Has a little 2 on it. And then to perform the add operation, I pop those two top plates off, create a 3, and put the 3 plate back on.

So that's how it performs computation. It's intrinsic to the notion of a stack machine.

Q. Sorry to interrupt. If you could go on --

A. Sure.

Q. -- with the rest of instructions and explain what they do in a stack machine?

A. Sure.

THE COURT: Can I ask you a question?

THE WITNESS: Yes.

THE COURT: You see that long list? It goes 0, 2, all the way up to 20. But it skips 1.

THE WITNESS: Yes.

THE COURT: Is there a reason for that? Or is that a goof-up? Why -- why does it have 0 to 20, but 1 is left out?

THE WITNESS: That's a good question, Your Honor. It's -- it basically -- the address and some instructions take more space than the others. So, for example, the dup instruction at address 3 takes 1 byte, and so we move to address 4. That's the next place that we can put something.

Imagine you're going down a street. Some people have a really big, fancy house with a wide yard, if you will. And then you get to another house that's very small. So the width of a property is a different size. And the same is true with the instructions.

The more information that you have to pack into these instructions, the wider they have to be. And so it's not counting instructions. That, in fact, would be 0, 1, 2, 3. Here we are moving by address of where we can put things in memory.

THE COURT: I don't understand why there's not a 1 there.

THE WITNESS: There is actually a 1 there. But the iconst_ 4 instruction occupies both address 0 and address 1.

THE COURT: All right. Now, I understand.

(Laughter)

THE WITNESS: I should have started with that.

THE COURT: We are going to break in about four or five more minutes. But continue on.

THE WITNESS: Sure. We can finish this slide.

Okay. So the second instruction says "newarray int." What this does is to allocate space by figuring out what the operand is on top of the stack, the 4. So it pops that value.

It then creates an array with four elements in it, four places where I can stick things. And, oh, by the way, the "int" means that -- they're integers, they're numbers. So then it pushes a reference to that array onto the stack. And so now I've replaced the 4 with a reference to an array.

BY MR. PAIGE:

Q. How big is the array, again?

A. It's size four integers.

Q. So it's got four places?

A. Four places to store four integers.

Q. Great. What is the dup instruction, number 3 do?

A. Here we make a clear distinction. We're moving from array creation time to array initialization time.

If, for example, we got rid of the "={1, 2, 3, 4}" then most of the rest of this code would disappear. However, for the purpose of initializing this static array, the Java compiler inserts a repeated sequence of bytecode instructions that individually assigns values to those spots in the array.

So, for example, we get to a dup. What that does is to duplicate the array reference that's on the stack. We execute iconst_0, which pushes a 0 onto the stack. That is the index. We start everything in computer science from 0. "iconst_1 at address 5 pushes a 1 on the stack. That is the value that we're going to store in position 0.

Finally, the store instruction pops three values off of the stack. It gets the array reference. It gets the index. It gets the value, 1. And now it stores that value at the appropriate position. And then we repeat the sequence: Dup push push store.

Would you like me to continue?

Q. Sure. What do the rest of those sequences do?

A. Well, they perform the same fundamental operation. The difference is that at 8 and 9 you see that I'm pushing a 1, which means I'm going to store value at index 1 in the array. I'm going to push a 2, which means I want the value of 2 to go into that position. I perform the store, again, which pops all of those elements off of the stack.

Remember, that's my scratch pad. I do everything on and off of that stack.

And then I store 2 at position 1. And without belaboring the point, at 12 and 13, and essentially 14, I store the value 3 at index 2. Continuing on, I store the value 4 at index 3. And, finally, I store the array that's left on the stack into the tester field, and return from clinit method.

Q. Now, all of those bytecode instructions are created by a Java compiler, right?

A. Yes.

Q. Does Oracle make the only Java compiler you can use for this purpose?

A. No. There are a number of others. ECJ, the Eclipse compiler. GCJ, the GNU compiler. Jikes.

MR. PAIGE: Your Honor, this would be a good point for a break, if you like.

THE COURT: All right. Fifteen minutes. Please remember the admonition.

THE CLERK: All rise.

(Jury out at 9:12 a.m.)

THE COURT: Please be seated. Any issues for the judge?

MR. VAN NEST: No, Your Honor.

MR. JACOBS: No, Your Honor.

THE WITNESS: Can I step down?

THE COURT: Yes, you can have a 15-minute break, too. We'll take our break.

(Recess taken from 9:12 to 9:30 a.m.)

THE COURT: Okay. Back to work. All set?

MR. JACOBS: I have something to raise with you.

THE COURT: All right. Do we need the witness here?

MR. JACOBS: No, we don't.

THE COURT: The witness should be excused.

MR. JACOBS: That would be better, I think.

THE COURT: All right. The witness should step out.

MR. JACOBS: Sorry.

THE COURT: Everyone else, be seated.

MR. JACOBS: And I mention this to give you a -- the Court a heads up about some things we may need to do as early as possible, even though I appreciate we may be calling in the jury in a second.

Mr. Van Nest and I continue to work on Phase Three issues, and we've agreed to advance -- I'm sorry, give me a minute.

(Pause.)

MR. JACOBS: We have agreed that both copyright willfulness and patent willfulness would be tried in Phase Two. And so the consequences of that are as follows: We will need to work on the instructions. We will need to work on the verdict form.

Mr. Van Nest and I have agreed as follows: Our current thinking is that the trial plan remains as is. He's going to do a few more things in his case in chief. We'll put on our very brief rebuttal case.

If I have predicted incorrectly what Mr. Van Nest is going to do in his case in chief, and I need to put on a more extensive rebuttal than I currently plan to do, then Mr. Van Nest would have an opportunity to respond to that rebuttal, on the issue of copyright or patent willfulness.

And then I'm just -- I think this is clear, in any case, we're talking about, obviously, willfulness to the extent infringement has been found so far in the case.

THE COURT: Okay. This may be a good way to streamline things, but are you suggesting that you will reopen your case to put in more willfulness evidence? Is that it?

MR. JACOBS: No. I think it is more in the nature of a rebuttal to what Google might do in its current defense case in chief.

And in order to assure Google that in our -- that if in our rebuttal we feel like we need to do a little bit more on willfulness than they currently understand we are going to do, they will have a chance to respond to that.

I think this is anticipating a non-issue, but it was something we were willing to do in order to accomplish this result.

THE COURT: All right. And are you proposing to adjust the hours? Maybe, you want to do that.

MR. JACOBS: I don't believe -- I think we have time, Your Honor. I think the -- the issue of schedule may be that just working out the instructions and the verdict form, even if we really hustle, we -- we may have a gap in the schedule. But we think that would be worthwhile as we continue to try and work through Phase Three issues and ultimately result in economizing.

THE COURT: Does that sound -- do you agree with everything I just heard?

MR. VAN NEST: Just about.

I think that we have an argument, that we need to discuss with counsel and the Court, that there is -- there is no remaining claim for willful copyright infringement based on what was found. But we'll work that out. Apart from that, I'm in agreement that we would try any willfulness issues that remain in this phase.

What Mr. Jacobs has told me was, he has a brief rebuttal planned. And all I've said is, if the brief rebuttal is what you've represented it is, that's fine. But if you're going to open up and do more willfulness stuff, then I should have a chance to respond to that. And he's agreeable to that.

So I still think it's likely we'll finish most of the evidence today. And, if not, we'll slop over a little bit to

Monday, but be in a position to close the evidence and do the closings, or part of them, on Monday.

That's our hope.

THE COURT: Well, what -- okay. Let's go back, then, to willfulness on copyright. The only two things that could possibly be in play are rangeCheck and decompiled.

MR. VAN NEST: That's right.

THE COURT: Right?

MR. VAN NEST: That's right.

THE COURT: So is plaintiff seeking willfulness as to those two files -- those two items?

MR. JACOBS: Yes.

THE COURT: So how do we deal with that?

MR. VAN NEST: Your Honor, willfulness is only an issue in copyright for statutory damages. Remember? That's our view, is, it enhances the statutory damages. If they are not seeking statutory damages anymore on the copyright side, then willfulness is not relevant on that side, and we'd be trying the patent willfulness alongside the willful blindness that we have now.

THE COURT: Let's find out. Is that the way the law works? Willfulness only screws into statutory damages?

MR. JACOBS: I'm going to defer to Mr. Boies.

THE COURT: Mr. Boies.

MR. BOIES: Your Honor, there's an issue on willfulness with respect to injunctive relief, but that's for the Court, not for the jury. We wouldn't have to have that for the jury.

In addition, we believe, although there is some split in the authority, that the better view is that willfulness is relevant to the issue of infringer's profits.

Now, to be clear about rangeCheck and these files, we are not going to produce a damages case. That is, we are not trying -- We don't believe, under the guidelines the Court has established, that it's possible to tie specific damages to rangeCheck or to these files.

The only issue is the issue of whether we are entitled to make an infringer's profits case. Not for all of their profits. We are not asking for all their profits. We are not asking for most of their profits. We may not even be asking for a large amount of their profits compared to the total profits that they make.

What we are saying is that, once you have proved infringement, we think under the law we have a claim for infringer's profit. And the burden of proof on that, under the statute, is on them, not on us.

The question is not how much we're entitled to. I think we probably would come maybe not too far apart on that. The question is, who has the burden there?

We're not going to have an expert report on this because we don't have one. We're not going to have proof of damages because we're not claiming damages. It's going to be an issue of infringer's profits.

Now, the Court may very well view this -- and I've taken the Court's comments to heart -- that given the relative contribution of these items to Android, there is an inability, as a matter of law, for us to seek infringer's profits.

If the Court views that way, we don't need a trial on the -- on these items because that's all we're seeking. However, if somehow the scales were to fall from the Court's eyes and the Court were to see that we are entitled to make an infringer's profits case, then under those circumstances copyright willfulness would be relevant.

THE COURT: Well, I've said you make your pitch to the jury. This is being sprung on me -- maybe not in one day, but certainly over a day and a half. And I don't want to make any rulings. I'm going to let you open to the jury, say you are going to ask them to award hundreds of millions of dollars for nine lines of code. I have a feeling that that won't help your case too much. But you are a great lawyer, you can do that. And the other side can say, no, how ridiculous that is.

And when somebody makes a proper motion, I'll make a ruling. But you're just alerting me now, and I'm not going to blurt out some ruling.

I have told you generally how I feel, but I -- I'm not saying to you, you cannot try. I said earlier to you this morning I took back what I said yesterday.

If you want to try to make out a case for infringer's profits based on nine lines of code and seven decompiled files that aren't even part of the Android system, you know, that's an extremely weak proposition. But until it gets briefed and all of that, I cannot tell you that it's illegal as a matter of law.

That's the way my instincts are. I will tell you that.

MR. BOIES: I can tell that's where your instincts were, Your Honor.

THE COURT: It could be that I would be surprised by some statement in a decision somewhere that would surprise me.

MR. VAN NEST: I don't think you'll be a bit surprised, Your Honor. I'm surprised to hear --

THE COURT: Well, then you can brief it for me.

MR. VAN NEST: That's what I'm going to do.

THE COURT: In due course, we will have a more orderly process. In the meantime, right now, the way things are we are going to have Phase Three.

So, see, your stipulation is falling apart because you can't agree on whether or not willfulness has any relevance on the copyright side.

So I'm okay with you if you want to submit willfulness to the jury. Whatever you both tell me you're willing -- you both have to agree, now, that willfulness on these issues will go to the jury. And by "these" I mean you define it. Copyright. Patent.

And if by consent you want to alter the pretrial order, that's -- that's good. I can go for that. But until you both do it clearly, and you know what you're agreeing -- and you say what you're agreeing to and you agree to it, then I'm not going to change things yet.

MR. VAN NEST: That's right. That's what we would request.

THE COURT: You need to say you have an agreement. And right now I see 95 percent agreement, but I don't see a hundred percent agreement.

MR. VAN NEST: Me, too.

MR. BOIES: I think that's exactly right, Your Honor.

THE COURT: All right. So we roll on. Let's bring back our witness and bring back the jury.

Question. Did you have that demonstrative on purpose so it looks like a rocket?

(Laughter)

THE COURT: Is that your idea, to remind the jury --

MR. VAN NEST: You know, I had that same thought when it came up, Your Honor.

THE COURT: Like Friendship 7.

MR. VAN NEST: I didn't stick anybody's face on there.

(Jury enters at 9:42 a.m.)

THE COURT: All right. Welcome. Welcome. Be seated, please.

The jury is ready for you to proceed. Go right ahead.

MR. PAIGE: Thank you, Your Honor.

BY MR. PAIGE:

Q. Dr. Parr, could you take a look at the '520 Patent, Exhibit 4011, please.

A. Sure.

Q. And, specifically, could you look at Claim 1 of the patent, which Oracle is asserting in this case.

A. Yes.

Q. And taking a look at Claim 1, could you explain to the jury what the steps in Claim 1 do.

A. Yes. The first step is, where it says, "Compiling source code containing the array with static values," that's talking about the Java compilation process that we demonstrated before the break, where we take human readable Java source code and compile it down to Java bytecodes.

The second step in Claim 1 says, "Receiving the class file into a preloader." That just means we're literally loading those bytecodes that we created in the compilation step, we are loading those into something called the preloader.

The third step says, "Simulating execution of the bytecodes of the clinit method against a memory without executing the bytecodes to identify the static initialization of the array by the preloader."

The fourth step is just storing into an output file an instruction for the purpose of initializing that array.

And, finally, the fifth step talks about a modified Java Virtual Machine that knows how to interpret that instruction.

Q. Let's talk a little about some of the terms of art you've just used in Claim 1.

What does the patent say about the clinit method you referred to? I might direct you to Column 1 of the patent, around line 57.

A. The patent is describing -- would you like me to read it or describe?

Q. Explain what it says here in the patent, for the jury, please.

A. So the clinit method stands for "class initialization method." Its job is, essentially, to initialize all of the so-called static elements within a class. Among those will be static array initializations.

Q. Now, in Claim 1 there was also a reference to simulating execution of the bytecodes, correct?

A. Yes.

Q. And could you explain to the jury what simulating execution of the bytecodes means.

A. Well, let's draw a distinction here. It says, "Simulating execution without executing the bytecodes." It's probably worth explaining what that means.

I understand that to mean -- uhm, "execution" to mean running live on the Java virtual machine.

Simulating execution is in -- is occurring before runtime in a preloader. You can think of this as a dress rehearsal for a live show. So that is the distinction I would probably draw. So the goal there is to simulate execution of these bytecodes to determine the static initialization elements of this array.

Q. Now, what's the core requirement that you need to have in order to have simulation of execution in a stack machine?

A. Well, there is no meaningful definition of simulating execution of a stack machine without manipulation of a stack. That means pushing, popping, things like that.

Q. Is manipulation of the stack a part of the simulated execution process described in the '520 patent?

A. In Claim 1 in step three it does not say stack manipulation.

Q. In Code Table 5 what does it refer to?

A. In the specification of the patent if we look at Code

Table 5, if we could highlight, say, line 42-ish? The definition of a stack. If we could highlight "object stack []" and the comment beyond it?

(Document highlighted)

A. So in the specification of the patent I see here an implementation of simulated execution. The first thing we see is the definition of a stack that they can use for simulating execution. And the comment clearly indicates that it's going to create that stack for the purpose of play execution or simulated execution.

Further, if we jump into the code itself, say, Line 56, '7, here this method is clearly operating on the stack. This is a push negative one instruction. The comment indicates that, push negative one on the stack.

If you look at the code, you don't have to worry about the details itself, but it says "stack of stack pop" -- That means push -- "equals a negative one." So here we're manipulating a stack.

And the other instructions you'll see, such as Line 60, is another stack manipulation. 63 and '4, stack manipulation. 65, 66, those are pop operations, and so on.

Q. Were you in court for Dr. Mitchell's testimony on the alleged infringement of the '520 patent?

A. Yes.

Q. Do you agree with Dr. Mitchell's conclusions as to infringement?

A. I don't agree.

Q. Why not?

A. I don't agree because the dx tool does not use simulated execution for the purpose of identifying static initialization of an array.

Q. Now, could you tell the jury how you reached your opinion that the Android dx tool doesn't do that?

A. Sure. I spent quite a bit of time reading through the source code, stepping through with a debugger, and I ran two experiments to confirm my hypothesis.

Q. What source code did you examine?

A. I examined the source code for the dx tool in the Android developer kit, I guess.

Q. And before we go into your examination of the source code, can you explain the purpose of the Android dx tool?

A. Yes. The dx tool takes the .class files that are emitted by the Java compiler and it translates them to .dex files.

Q. And why does it do that?

A. Well, because the Java Virtual Machine and the Dalvik Virtual Machine have completely different instruction sets. So a translation has to occur, otherwise you have a situation as if you had a person who only spoke German trying to give instructions to someone that only understood Portuguese.

Q. So, but we've heard that Android applications are written in the Java programming language. How can such programs run on the Android Platform if Android uses a Dalvik bytecode?

A. Well, that's really the purpose of the dx tool. It performs a translation of the original Java bytecodes into Dalvik bytecodes so that it can execute.

MR. PAIGE: May I approach, your Honor?

THE COURT: You may.

(Whereupon, document was tendered
to the witness.)

BY MR. PAIGE:

Q. Dr. Parr, I'm handing you Exhibit 46.16. Can you tell me what that is, sir?

A. This is a file called simulator.java. It looks like it comes from the Froyo distribution. It is a class I recognize from the dx tool.

MR. PAIGE: If we could put up Page 1 of that exhibit, please?

(Document displayed)

BY MR. PAIGE:

Q. Looking at Line 37.

A. Yes, I see that.

Q. It says there a comment.

"Class which knows how to simulate the
effects of executing bytecode."

A. Yes.

Q. And this is part of the dx tool, correct?

A. Yes.

Q. Now, apart from identifying the initializing of a static array, how does the Android dx tool translate these .class files into .dex files?

A. Well, other than for the purpose of identifying static initialization of an array, the dx tool uses simulated execution to perform the translation.

Q. And what does it do when it uses simulated execution?

A. Basically what it does is to observe the side effects of these instructions and re-expresses them as Dalvik bytecodes.

Q. And how does it determine the side effects of the instruction?

A. It pushes things on the stack. The dx tool observes these, recognizes them as operands for instructions, converts those stack elements to registers.

Q. And so you would agree with that comment, the simulator class does know how to simulate the effects of executing bytecode, correct?

A. I would agree. It would call it simulating execution, but that's fine.

Q. So if the Android dx tool simulates execution of instructions from Java bytecode in order to translate them into the dex Dalvik bytecode, why doesn't it infringe the '520 patent?

A. That's because for the very specific purpose of identifying the static initialization of an array, it does something different. It does pattern matching.

Q. Well, what would one of ordinary skill in the art understand the phrase "identify the static initialization of an array" to mean?

A. In simple terms it just means to find the values that should be the default values for that array.

MR. PAIGE: May I approach, your Honor?

THE COURT: You may.

(Whereupon, document was tendered to the witness.)

BY MR. PAIGE:

Q. Professor Parr, I've just handed you Trial Exhibit 46.17.

(Document displayed)

Q. What is that exhibit, sir?

A. This is a file called BytecodeArray.java. Again, it appears to be from the Froyo distribution. This is part of the dx tool.

Q. And this is source code you had looked at before?

A. Yes.

Q. This is something the jury will have back in the jury room with them, so it's useful to look at the complete code rather than just snippets.

Where does that source code come from?

A. This is from the dx tool.

Q. Could you please tell us what you found in that source code that's relevant to your opinion?

A. What I found -- well, what I identified was the specific part of the program, the dx tool, that identified the static initialization of the array. And that is in a method called parseNewarray.

If we can put up Line 887, please?

(Document displayed)

A. So this is the method in which the identification occurs. So, actually, I guess the easiest thing to point out is if we turn to Line 948.

(Document displayed)

A. There is a comment there written by the programmer. It says:

"Try to match the array initialization idiom."

Q. What is an idiom?

A. Pattern.

"For example, if the subsequent code is initializing an int array, we are expecting the following pattern repeatedly."

And here we see the sequence, the pattern that I've identified in my first slide: Dup, push, push, store. That initializes an individual array element.

So I looked below to see what the code did and I found it consistent with that comment. To me, it appeared to be a classic example of a pattern matcher.

Q. Why is that?

A. Well, I have been building parsers for 30 years and it was very easy to see this.

But to point out a couple of the key elements here, you see on Line 965 there is defining a variable called "punt," which means another word for give up, a colloquial term for that. And as you step through the code, you'll see that it says if I don't find a dup, on line 69, then punt. Basically break.

And it says "punt" in a couple of places down here, but the thing is, it's looking for something, it doesn't find it, fail. If it's looking for the next thing and it doesn't find it, fail.

So this is a classic example of that pattern recognition.

Q. So is it manipulating the stack at all as it goes through this set of instructions?

A. No. I didn't see any from the source code.

Q. Now, you referred some comments in the code. Do the comments in the code control what the code does?

A. No.

Q. How are they useful here then?

A. Comments are meant to aid other developers or, indeed, our future selves since these programs are complicated. So it's a quick way to understand the intention of the original author.

Q. Did you find the comments to be consistent with what the code actually does?

A. I did.

Q. Now, you mentioned you conducted some experiments, as well as looking through the source code.

What experiments did you conduct to test your hypothesis that identification of a static initialization of the array was accomplished through pattern matching?

A. Well, the first experiment was a simple test to see if I was correct, that there were no stack manipulations within the identification of the static initialization.

So, really, I just put print statements in the code that manipulates the stack. So there are methods called, you know, push, pop. And they're kind of like alarms. So if something popped up, "Hi, I'm in push." That means, okay, I triggered it. I'm doing a stack manipulation.

Q. Have you prepared a demonstrative showing your tracing of this code?

A. Yes.

Q. Could we put up Slide 2, please?

(Document displayed)

Q. Could you explain to the jury what your demonstrative here shows? For example, what does it mean to "enterBytecode Array.parseNewarray"?

A. Well, let me back up two seconds here and explain a little bit of context. So all of these lines -- TJP are my initials. And TJP19, that's my age... Okay.

Now, these are all log statements that tell me what is going on in the program. So kind of like the print statements in the stack area, they give me an idea of the flow of the program. It says, "I'm here." "I'm here."

It's kind of like you're walking through the woods and you radio home to say, "Hey, I'm at this fork in the woods. I'm at this point."

And now if you take a look at line TJP21, you'll see it says, "stack push5." So during the normal operation of the simulated execution we see stack manipulation.

As we enter the parseNewarray method, so line TJP23, is a log statement that indicates the program has entered method parseNewarray. What I observed was that the stack alarms went silent until the parseNewarray method had exited.

So if you look through 23 through lines 35, there are no stack alarms; but once we return from parseNewarray, then we get, you know, once again, stack manipulations.

Q. So what did that tell you?

A. Well, it told me that since there are no stack manipulations, that it cannot be using simulated execution to identify these initialization elements.

Q. Thank you.

Now, what was the second experiment you performed to confirm your hypothesis?

A. Okay. This one is a little more tricky, but it demonstrates things very well. I was looking for an experiment that would clearly differentiate between pattern matching and simulated execution. So the idea that occurred to me was if this sequence, this sequence of templates of patterns that the parseNewarray is looking for, if that sequence is altered, it should break the pattern and it should not generate an instruction to initialize the array in one fell swoop.

However, if it were simulating execution, then a minor tweak wouldn't cause a problem. It's kind of like in English, if I said, "I like apples." One of my students would say instead, "I totally like apples." And simulated execution in my brain understands the meaning. However, the pattern recognized by an English teacher's grammatical structure would not match that as grammatical.

So, can I draw?

MR. PAIGE: Your Honor, may Dr. Parr use the easel?

THE COURT: Sure.

(Witness steps down.)

THE COURT: Be sure to press pretty down hard on most magic markers because everybody in the jury box needs to see it clearly.

THE WITNESS: Okay. Is there a color that you can read better? Black --

THE COURT: Black is better.

THE WITNESS: Okay.

THE COURT: Assuming it has plenty of ink.

MR. VAN NEST: It's a brand new one I got yesterday, your Honor.

THE COURT: Thank you.

(Witness drawing on the easel.)

A. So let me give a little bit of background to show you precisely what this compiler is going to do, and then I'll tweak it and we're going to see the effect.

Okay. What I have here are the initialization elements of an array. This is a human readable code that the compiler consumes and uses to generate bytecodes. What do these bytecodes look like? Well, let me refresh your memory. I have an "iconst3" instruction. This pushes 3 on the static. This is our little scratch pad.

Then you'll remember our friend the Newarray instruction, which has an argument of int. That just means there are numbers in this array. So when I push a 3 on the stack, this Newarray instruction is going to pop that off and say, ah-hah. You want an array with three elements.

So, for example, I can build one over here. There I have an array with three elements in it. And I'll point out that the Newarray instruction also zeroes that out. So they are all zeros.

All right. Now we're done with the array creation and we're going to move on to initialization.

Remember, in computer science they love to start with zero. So position 0 is the first place. Position 1, position 2, or index. So I need to fill each one of these individually, and I'm going to do that with a sequence of instructions for each element.

First thing I need to do is duplicate the top of the stack which has the array. So it's my scratch pad, and I'm going to just get a reference to that array. Now I'm going to push the index, which is 0. I'm going to push the value, which is 1 that I want to store. I then use the store instruction to take those elements off of the stack and use the information to put a 1 at the first position.

And so this is the pattern I'm talking about here: Dup, push, push, store.

So, similarly, I'm going to initialize the second element in index one and I'm going to put a 3 there because that's my second number.

So I begin my scene again: "Dup," which duplicates the array reference. "iconst_1," because I want to put something in index one. And "iconst_3," that's the value.

"iastore," takes all these off of the stack, puts a 3 there.

And just for completeness, hard to stop mid-stream, let's finish it off.

Now I'm pushing a 5, which is the value to store in the last place, and I perform the store instruction. And now I have a 5 over here (indicating)

So does that give the background you're looking for.

Q. Yes. And then at that point what did you do to change the pattern?

A. Oh. Well, first let me say that I ran the dx tool on this sequence that is generated by the compiler. And what I found that it, indeed, creates an instruction to initialize this array all in one go. So the default output of the compiler is recognized by the dx tool as static initialization.

Okay. So I want to tweak this. I don't want to change the end result. I want 1, 3, 5 (indicating).

So how can I change this sequence of instructions without modifying the end result? Well, as I mentioned, the new array puts zero everywhere. So the array has a zero already. So I'm simply going to do nothing by putting another zero there.

So I essentially cut and paste this first bit of code and I say "dup," our friend "iconst," index "0," value "0," "iastore." And so this is a tweak that does not affect the code. If this executes, it wouldn't change the end array.

However, if the dx tool is a pattern matcher, it will fail to recognize this pattern because it turns out that part of the sequence is this number, the index, must be followed exactly by the next index, which must be followed by the next index. And I've broken that scene instead of going 0, 1, 2, I said 0, 0. At that point the dx tool should punt. It should fail.

So I ran the dx tool on this modified bytecode stream and the dx tool failed to generate an instruction to initialize these elements in one go.

Q. And, Professor Parr, if you could show the slides that you prepared showing exactly what you had done in your experiment?

A. Oh, sure.

(Witness resumes stand.)

MR. PAIGE: Could we bring up Slide 3, please?

(Document displayed)

BY MR. PAIGE:

Q. What's on the left-hand side there?

A. Okay. Without my messy handwriting, on the left-hand side we see something that is akin to what I've drown on the board.

The difference here is that since I know you've all memorized bytecode instructions now, it's storing 5, 4, 3, 2, 1 at indexes 0, 1, 2, and so on.

Okay. So that is the original bytecode generated by the Java compiler.

On the right we see the useless instructions, the instructions that don't change anything because zero is already there. That is the modified sequence.

Q. And these two things are actual pieces of code that you ran and put in your report in this case, right?

A. Yes, I did.

Q. When you ran the left-hand side, the original sequence, do you have a slide showing what that produced by the dx tool?

A. I do. If we could move on to my next slide?

(Document displayed)

Q. What does this show, Professor Parr?

(Document displayed)

A. These are the Dalvik bytecode instructions created by the dx tool in response to the original bytecode sequence generated by the Java compiler.

Q. And could you point out the instruction that generates the array?

A. Yes. "fill-array-data" on the third line below where it says "line 2."

Q. And could you show what the result was when you put your modified code into the dx tool?

A. Sure. If we can turn to my next slide?

(Document displayed)

A. There we do not see a fill-array-data instruction to initialize that array in one go. You see the normal conversion of Java bytecodes, stack code, to Dalvik instructions, which are register instructions. All of that v6, v5, all that, those are registers.

Q. If the dx tool were, in fact, simulating the execution of that modified Java bytecode, what would have resulted?

A. If the dx tool were simulating execution inside parseNewarray, it wouldn't care about the extra instructions.

Think of it this way. Imagine we want to match a pattern of any two numbers added together, 1 + 2. I can identify that pattern as 1 + 2, or 3 + 4. A functionally equivalent expression is 0 + 1+ 2 = 3. However, it breaks the pattern of any two numbers added together.

So maybe that helps to explain the difference.

Q. That in mind, what element of Claim one, if any, is not found in what the dx tool does, in your opinion?

A. Well, step three of Claim 1 specifically requires simulated execution of these bytecodes. And so it cannot infringe because the dx tool does not simulate execution for the purpose of static array initialization.

Q. And does Claim 20, the other claim that Oracle is asserting in this case, have analogous elements?

A. Yes, it does. It has --

MR. PAIGE: Can we put up Claim 20, please?

(Document displayed)

A. Right. The step there:

"Simulating execution of the code to identify
the static initialization of the array."

Q. So, in your opinion, does the dx tool infringe Claim 20?

A. My opinion is that it does not infringe.

Q. Were you present in the court a few days ago for Mr. Poore's testimony?

A. Yes.

Q. Now, we heard Mr. Poore talk about space savings that would result from the substitution of the fill-array-data instruction.

A. Yes.

Q. Do you have an opinion as to Mr. Poore's conclusions?

A. Yes. I disagree with his conclusions because his tests were not on real world applications.

Q. Well, why doesn't his analysis provide insight into the impact that use of this method would have in the real world?

A. Simply because his tests were contrived examples that were not meant to be real world examples.

Q. Why does that matter?

A. Well, the impact of a patent, you know, you want to see how it is affecting real world applications because that's where the value is, I guess.

Q. Did you perform your own analysis of the potential impact of this functionality?

A. Yes, I did.

Q. What did you do?

A. I analyzed about 20 applications, real world applications for the Android phone, to see what the impact was of generating this fill-array-data instruction.

So by analyzing these applications, I determined how many arrays were used in the original program, how big the space was needed by these arrays, and then computed a percentage that indicates what percent of the .dex file is consumed by these instructions.

Q. Where did you get the applications you used from?

A. I got them from counsel.

Q. Okay. And have you prepared a demonstrative slide showing the results you found?

A. Yes.

MR. PAIGE: Could you show that, please?

(Document displayed)

BY MR. PAIGE:

Q. Could you explain what's on this slide here?

A. Yes. In the first column I placed the name of the application that I examined.

The second column is the size of the Dalvik executable file, the .dex file. That's in bytes.

The third column shows how many static arrays were in that dex file.

The fourth column shows just how many bytes were in that array or arrays.

And the final column shows you the ratio, the percentage of the bytes in the dex file that are consumed by these arrays.

Q. And about how much of these dex files was taken up by the arrays?

A. Well, as you can see from the right column, it's an insignificant part of the dex file. And, in fact, if you look -- I don't know how many exactly -- there are seven, eight of these real world applications that don't have any static arrays.

Q. Can you name some of those?

A. Sure. GoogleBackupTransport, GoogleCalendarSyncAdaptor, GoogleContactsSyncAdaptor, GoogleFeedback, and so on.

Q. Given that finding, what's your opinion as to the impact of the accused functionality on the size of ordinary Android applications?

A. Well, from my own experience developing Java for 17 years, and from examining these applications, I can tell you that these are just not a problem.

Q. So to sum up, if we can put up Slide 7, what's your opinion on infringement of the '520 patent and why do you hold that opinion?

A. Well, my summary is that the dx tool uses pattern recognition or pattern matching, not simulated execution. The reason being it does not manipulate a stack while it identifies these static initialization array elements, and so it cannot infringe '520 patent Claim 1 nor Claim 20.

MR. PAIGE: Thank you, Professor Parr. I have no further questions.

MR. VAN NEST: Your Honor, can we stick an exhibit sticker on this demonstrative?

THE COURT: Yes, you can. Now, just to remind the jury -- what number will that be.

MR. VAN NEST: It will be Trial Exhibit 3543.

(Trial Exhibit 3543 marked
for identification)

THE COURT: It's not in evidence. It's just going to be marked for the record. It's not going to be in the jury room, so if there is something on there of interest to the members of the jury, they will make notes in their notepads, please.

All right. Now it's time for cross-examination.

CROSS EXAMINATION

BY MR. JACOBS:

Q. Good morning, sir.

A. Good morning. I feel like I know you now after seeing you for five days.

Q. I think there are several people in the courtroom who feel that way.

(Laughter.)

Q. I guess I don't understand exactly what you're saying. Are you saying that the accused functionality in the dx tool is valueless?

A. I'm saying that its overall impact on real world applications is insignificant.

Q. Insignificant.

A. Yeah. I'm not saying -- it's greater than zero, but not zero.

Q. But not?

A. Not zero. Some applications don't use static arrays. However, in general statically speaking, it does not affect the overall size in any significant way for these real world applications that I evaluated here.

Q. So the Google developers developed functionality in the dx tool that is of near zero real world significance?

A. Well, you know how we programmers are fairly precise. And especially when you get, you know, as old as I am you remember machines that had 16k ram. You can't take that efficiency demand out of us.

And so it looks like easy code to write, so it seemed like an easy win. And I can only assume -- I don't know what their thought process was, but I can only assume that they were responding to their normal instinct.

Q. And when Mr. Bornstein spoke at the Google IO conference -- you have seen that testimony in the courtroom -- and talked about saving 100k of memory, you think he was just being instinctive?

A. We like the idea of saving things, and so I assume that his instinct was, yes, that's good.

I note that it's -- I don't remember how big Mr. Bornstein said the libraries are. But did he say it was 10 megabytes, 20 megabytes? I can't remember. 100,000 versus 10 million.

That's insignificant -- not insignificant, let's say, but in that particular special case, you know, not huge.

Q. And it's still in the dx tool today; true, sir?

A. I would assume so.

Q. For Android 2.3 and predecessor versions, the versions that are at issue, it was never removed?

A. I believe that's true.

Q. And your testimony is that if it had -- if a developer decided, you know what? I need to use -- I don't need this code any more. I think I should just take this code out, it would have an insignificant impact on real world performance of the phone?

A. That's my observed computations here, yes.

In fact, I have to look back at my report, but I think that one of the Google engineers told me that this was not in the code originally.

Q. So it wasn't in the code originally and then got added?

A. I believe that's true.

Q. So now it's not kind of just instinctive, but somebody makes an affirmative decision to add it to the code? Insignificant real world, in the real world?

A. Sure. Basically what happens is you develop software not in one, you know, big lump. You do the high priority items, and then you eventually get to the low priority items.

Q. And your testimony is that this must have been a low priority item for the Google developers?

A. Seems reasonable. I don't know what they were thinking.

Q. But your testimony today is -- at least as of today is this is of near zero real world value?

A. That's what it looks like to me with the .01, .02 percent numbers, yes.

Q. And that doesn't call into question your choice of applications to test this functionality?

A. I was given them from counsel. I cannot attest to -- you know, I don't think they cherrypicked these, if that's what you're suggesting, but I cannot tell you where they got them.

Q. And you didn't make your own independent choice of applications to test static initialization of an array in the dx tool?

A. I tried. Unfortunately, I have an iPhone. I should say, fortunately, I have an iPhone. I prefer the iPhone, so I don't have an Android phone. And I specifically went to the Android store to download the top 20 applications but, of course, it said, you know, "Put in your phone ID," or something.

Q. So counsel gave you the applications to test?

A. Correct.

Q. And counsel gave you the applications to test that are reflected in the report and in the testimony you gave in court today?

A. Yes. Although I can only imagine that Gmail, Street, Talk, Maps, YouTube, I can only imagine that those are fairly popular applications and real world.

Q. And the Android code itself you did not test; true, sir? You did not test static initialization of arrays in the Android code?

A. Could you define "in the Android code" for me?

Q. Meaning, the Android code base.

A. I did not take the entire source base and perform an analysis on that tool, no.

Q. And so there are two possibilities. One is that your applications were not real world. Another possibility is that the focus of this technology isn't the applications, but it's the platform.

In either of those cases missed a possibility of the -- possible value of the '520 patent; true, sir?

A. If I understand what you're saying, and the applications given to me by the attorneys such as Gmail, Street, maps, youTube are not real world, then I would have been not using real world applications; but I believe that they are, in fact, real world applications.

Q. I think we all agree that those are real world applications. The question is whether there are other real world applications or the Android platform itself that depend heavily on static initialization of array? Would you agree with me that that's a possibility here?

A. Oh, yes. You can make a program that is mostly arrays, which is what I saw in the Poore report, for example.

Q. And you could make an application program that was heavily array generating; true, sir?

A. Yes.

Q. A real world application, a useful application?

A. Yes.

Q. And you didn't test such an application?

A. I tested many applications, but I did not test one that was purely static initialization.

Q. You did not test real world applications that are heavily dependent on static initialization functionality?

A. That is correct.

Q. And counsel could have given you an Android phone, of course, so you could have gone to the app store and picked your own applications; true, sir?

A. I believe so.

Q. Let's see if we can focus on where -- a little bit more closely on where we are in agreement and disagreement.

MR. JACOBS: Can I have the Elmo?

A. May I follow up here your line of questioning with a comment?

BY MR. JACOBS:

Q. I'll let you do it on your counsel's -- your counsel now knows that he better ask you a question or you're going to be upset with him.

(Document displayed)

Q. This is Claim 1 of the '520 patent. And according to your testimony, sir, there is no dispute that we have in the dx tool:

"...a method in the data processing system
for statically an array."

True, sir?

A. Yes. There is a method within the dx tool to identify the static initialization of the array. The end result is not in question.

Q. The end result is not in question; correct, sir?

A. That is correct.

Q. And then in the next step we -- you acknowledge -- or, you did not testify that this -- this limitation is not present when a developer creates an application; that is, that:

"The developer compiles source code
containing the array with static values to
generate a class file with a clinit method
containing bytecodes to statically initialize
the array to the static values."

That limitation is present when developers develop applications using the Android SDK; true, sir?

A. Yes. The developer would have to use the -- a Java compiler to generate those instructions.

Q. We're not disputing that a class file is received into what represents a preloader in the Android environment; true, sir?

A. True.

Q. And skipping the one in red, so we can focus afterwards on the one in red.

There is no dispute that after the static initialization, after the red portion of the claim, that:

"The output file includes an instruction
requesting the static initialization of the
array and that instruction is stored."

True, sir?

A. There is no dispute that the dx tool stores an instruction.

Q. And there is no dispute that in the Android environment that instruction is:

"...interpreted by a virtual machine to perform the static initialization."

True, sir?

A. That is true.

Q. Okay. Now, let's focus in on the red, and I want to work it backwards because I want to really hone in on where the disagreement is.

A. Uh-huh.

Q. You agree that there is:

"...an identification of the static initialization of the array in the dx tool."

True, sir?

A. Yes.

Q. So if we -- we could actually -- just for -- I understand that you're objecting to the whole clause, but just to really hone in let me just shade these out for a minute.

But so far we have established that there is:

"...an identification of the static initialization of the array in the dx tool."

True, sir?

A. We agree that the dx tool identifies static initialization of an array.

Q. We also agree that it:

"...does so without executing the bytecodes."

True, sir?

A. I interpret "executing the bytecode" in this context to mean on the life Dalvik VM or Java VM.

Q. And that's not how it's done and so, therefore, you would agree with me that this is done without executing the bytecodes in the dx tool; true, sir?

A. In this context, yes.

Q. And you also agree that what we're talking about when we're doing the function in the dx tool is:

"...directed to the bytecodes of the clinit method against a memory."

True, sir?

A. Yes. The static initialization of an array is done with instructions in the clinit method.

Q. So actually the only issue from a technical perspective is what Android does is simulate execution of the bytecodes, true, sir?

A. No, that's not entirely correct. It is correct to say that our issue is that the dx tool does not use simulated execution for the specific purpose of static initialization.

Q. Fair enough.

So we agree that there is elsewhere in the dx tool simulated execution of the bytecodes; true, sir?

A. That is true.

Q. But for -- and that's the reason -- I didn't mean to suggest there was anything misleading by the underlining here.

A. Sure.

Q. But that's why the underlining goes all the way to "static initialization of the array" because your contention is that the dx tool does not simulate execution for that purpose; true, sir?

A. Or the purpose of static initialization, it does not use for simulated execution, correct.

Q. And just to hone in on it, the reason is, according to your testimony, recognizing a pattern is not simulating execution of the bytecodes without executing the bytecodes?

A. It is my contention that the dx tool uses pattern recognition as I've described there to identify the static initialization of an array.

Q. And that that technique does not represent simulating execution of those bytecodes?

A. That's right. It does not represent simulated execution of those bytecodes.

Q. Because, according to you, that has to be done in a stack machine against a stack; true, sir?

A. It has to manipulate a stack -- the only meaningful definition of using simulated execution on a stack machine is to manipulate a stack.

Q. And that's the -- really, the core of your testimony is that the technique in the dx tool does not use the stack for pattern recognition; true, sir?

A. We have to be careful when you say "the stack," because there's a number of things going on here.

But the dx tool does not manipulate the Java stack, if you will, while it determines these static initialization elements.

Q. And even though the word "stack" appears nowhere in this claim, nowhere in the red underlining, it's your contention that simulating execution of the bytecodes as represented in this limitation requires use of the stack; is that true?

A. That is true, and the reason I --

Q. I'm sorry, sir. Your counsel will have a chance.

A. Oh, sorry.

Q. Now, you looked at 46.16 with Google's counsel. Let's just go there again.

A. Okay.

Q. And let's look again at line 37.

(Document displayed)

A. Our favorite line.

Q. Yes, probably my favorite. I'm not sure it's your favorite.

A. Oh, sure.

Q. So line 37 is the line of code which says that:

"This class knows how to simulate the effects of executing of bytecode."

True, sir?

A. I would say that it knows how to simulate execute the bytecodes, but yeah.

Q. And that's part of the comments in the source code; true, sir?

A. True.

Q. And you heard the engineer who wrote this line say that he strove to write comments that were of high quality. Do you recall that testimony?

A. I heard two engineers say that. Which one?

Q. Both, Bornstein and McFadden said that.

A. Yes. Then one of them did then for sure.

Q. So you would agree that this file simulator.java is a class that simulates execution of bytecodes?

A. Yes. It simulates and executes bytecodes.

Q. Now, let's go to line 110 near the bottom of the second page.

A. Okay.

(Document displayed)

Q. Line 110 the word "simulated" appears again. Do you see that, sir?

A. Line 110. "Simulates," I see that word.

Q. (As read)

"Simulates the effect of the instruction at
the given offset by making appropriate calls
on the given frame."

Do you see that?

A. I do see that.

Q. So that's yet another engineer comment about the source code in which "simulation" is the word that the engineer chose to use; true, sir?

A. That is true, although I will point out that --

Q. Sorry, sorry. Just...

A. Okay, sorry.

Q. Now, if we go to Line 117 to 119 on the top of the next page.

(Document displayed)

Q. That's a method that:

"Simulates execution of a single bytecode instruction."

True, sir?

A. If that were to be executed, yes.

Q. I'm sorry. I didn't hear what you said.

A. I'm sorry. If that were to be executed, yes.

Q. What's the -- is there a reason for the "if," sir?

A. My evaluation of the software indicated that method was never used.

Q. That method is in the code, but never used?

A. Correct.

Q. And is that anywhere in your report, sir?

A. No. I found that a few days ago.

MR. JACOBS: Move to strike, your Honor.

THE COURT: No, it's a cross-examination. You asked the question. It comes in.

BY MR. JACOBS:

Q. In order to simulate execution of a bytecode instruction, that code would make a call to another method called parseInstruction on line 119; true, sir?

A. True.

Q. So in order to simulate execution of a bytecode instruction using this code, you need to parse the instruction; true, sir?

A. I would use the term "decode," but, yes, I understand what the developer meant.

Q. So parsing the instruction here is part of simulation execution?

A. Parsing a single instruction, yes.

Q. So let's go back up to Line 86 on Page 2.

A. Okay.

(Document displayed)

A. Yes.

Q. It says:

"Simulates the effect of executing the given basic block."

Do you see that?

A. I see that.

Q. And that's another engineer comment about the source code; true, sir?

A. Yes.

Q. And this block of code here beginning on line 92 is a method that simulates execution of a block of bytecode instructions; true, sir?

A. True.

Q. And, again, to simulate execution of the bytecodes, the method makes a call to a method calls parseInstruction that's on Line 99; true, sir?

A. True.

Q. So, again, in order to simulate execution, this case of a block of bytecode instructions, you parse the instruction; true, sir?

A. It individually parses an instruction, yes.

Q. You need to parse the instruction; true, sir?

A. That is true.

Q. Now, when the dx tool receives these long lists of bytecodes to initialize a static array, you agree that simulator.java will visit at least some of those bytecodes; true, sir?

A. I'm not sure I understand your question. Can you rephrase?

Q. Why don't you look at your report at Paragraph 71.

A. Okay.

Q. You wrote there:

"The code in the simulator class visits the first two bytecode instructions related to static array creation."

A. Yes, I see that.

Q. (As read)

"...but does not touch another instruction until the final two bookkeeping instructions."

Do you see that?

A. Yes.

Q. So to ask you again: It is true that simulator.java when it receives a long list of bytecodes to initialize a static array will visit at least some of those bytecodes?

A. It will also --

Q. Sorry. "Yes" or "no," sir?

A. Oh, sorry.

As part of creation of an array and initialization, yes.

MR. JACOBS: No further questions.

THE COURT: Anything more by Google?

MR. PAIGE: Yes, your Honor.

REDIRECT EXAMINATION

BY MR. PAIGE:

Q. Dr. Parr, you got some questions from Mr. Jacobs at the beginning of his cross-examination about the applications that you looked at, correct?

A. Yes, I did.

Q. Now, you're aware that Dr. Mitchell submitted a report replying to your report, correct?

A. Yes.

Q. In that report did Dr. Mitchell analyze any applications himself?

A. Thinking back to when I last read that report, I don't recall seeing --

THE COURT: All right. He doesn't recall.

A. If you could show me the report, I'll look through it real quick.

THE COURT: Is this something you once knew?

THE WITNESS: I'm sure I did when I read through it, but I may be confusing it with the Poore report, your Honor.

MR. JACOBS: It is the reply report, I believe we're talking about.

MR. PAIGE: It is.

May I approach, your Honor?

MR. JACOBS: So I'm not sure where in the sequence we are, your Honor.

THE WITNESS: I do not remember seeing an analysis of real world applications by Professor Mitchell. So my opinion is that I did not see them there. They are not there.

THE COURT: What is the point you're trying to get at?

MR. JACOBS: I withdraw the objection, your Honor.

THE COURT: All right. Go ahead.

If he can remember by refreshing his memory, you may go ahead and proceed.

MR. JACOBS: I'm sorry. I withdrew the objection in light of the witness's testimony. If we're going to go down this path --

MR. PAIGE: Fair enough. We'll move on at this point.

THE COURT: All right.

BY MR. PAIGE:

Q. Dr. Parr you looked at or you were shown some parts of the simulator class. If we could bring that up, please? 46.16?

(Document displayed)

Q. Now, when you have the simulator class and it uses parseNewinstruction or parseInstruction, which I believe is on the second page of this exhibit?

A. Uh-huh.

Q. What does that parseInstruction do?

A. It decodes the individual bytecodes. As your Honor asked me earlier, why does it skip from zero to two? That's because it must occupy two positions. And parseInstruction decodes whether it's one byte, two bytes, things like that and what they do.

Q. And does parseInstruction itself lead to another class?

A. Yes. By calling that method, you are calling a method on a class called bytecode array.

Q. And is that 46.17 that we looked at earlier?

A. It is, yes.

Q. So the parseInstruction calls out of the simulator class into a new class, right?

A. Yes.

Q. And is parseNewarray, is that class -- or is that method called by things other than the simulator?

A. Yes. It's used by at least two things that I observed. One is called the basic block identification. I don't remember the exact method name. And the second I will think is called dex dumper.

Q. And do either of those have anything to do with the simulator?

A. No.

Q. Now, if I could turn to Paragraph 71 of your expert report, which Mr. Jacobs has just asked you about.

MR. PAIGE: Could we bring that up, please.

MR. JACOBS: Your Honor, I object to this being displayed to the jury.

THE COURT: What's the objection? Not in evidence?

MR. JACOBS: Not in evidence.

THE COURT: Well, if it's not in evidence -- what is this?

MR. PAIGE: This is what Mr. Jacobs has questioned him on. I don't mind if it's not shown to the jury. I would just like to clear up what Professor Parr was being asked about.

BY MR. PAIGE:

Q. Mr. Jacobs asked you about visiting the first two bytecode instructions?

A. Yes.

Q. Did those instructions identify the initialization of the array?

A. Those two first instructions do not.

Q. What do they do?

A. Those are part of -- well, those are the instructions for creating the array.

Q. Thank you, Dr. Parr. No further questions.

THE COURT: All right. Can the witness step down?

MR. JACOBS: Very briefly, your Honor.

THE COURT: All right. Go ahead.

MR. JACOBS: Elmo, please.

(Document displayed)

RECROSS EXAMINATION

BY MR. JACOBS:

Q. Well, this is your slide; correct, sir?

A. Yes.

Q. And right at the very top we're talking about creating a simulator and then simVisitor, stack push, simVisitor; true, sir?

A. Yes.

Q. Thank you, sir.

THE COURT: All right. Now may the witness now step down.

MR. PAIGE: Yes, your Honor.

THE COURT: Thank you, sir. You're free to go. Counsel will take care of those documents. Just leave those. Have a good day.

THE WITNESS: Thank you.

(Witness excused.)

THE COURT: Next witness.

MR. VAN NEST: Your Honor, I think at this time Google will call Dr. David August.

THE COURT: As in the month?

MR. VAN NEST: Yes, A-u-g-u-s-t.

DAVID AUGUST,

called as a witness for the Defendant herein, having been first duly sworn, was examined and testified as follows:

THE WITNESS: I do.

THE CLERK: Thank you.

THE COURT: Welcome. And you need to get about this close. The base will move around.

THE WITNESS: How is that?

THE COURT: Say your name.

THE WITNESS: David August.

THE COURT: I think that works, good. Okay. Counsel?

MR. KAMBER: Thank you, your Honor.

DIRECT EXAMINATION

BY MR. KAMBER:

Q. Good morning, Dr. August.

A. Good morning.

Q. Can you briefly describe your educational background for the jury please?

A. Sure.

(Brief pause.)

A. Can you repeat the question?

Q. Sure. Can you briefly describe your educational background for the jury?

A. Sure. In 1993 I earned a B.S. in electrical --

THE COURT: You're not talking into the mic. You can look at the jury, that's cool, but you've got to move the mic so that it adjusts to follow your voice.

THE WITNESS: How is that?

THE COURT: I don't know. Can you all hear?

THE WITNESS: I'm sorry, your Honor.

THE COURT: The jury is signalling you to speak more into the mic.

A. Okay. In 1993 I learned a bachelor of science degree in electrical engineering from Rensselaer Polytechnic Institute.

In 1996 I earned a Master's degree in electrical engineering from the University of Illinois at Urbana Champaign. And in 2000 I earned a PhD from the same institution, also in electrical engineering.

Q. Where are you currently employed, Dr. August?

A. I'm a professor of computer science at Princeton University.

Q. How long have you been at Princeton?

A. Since 1999.

Q. What positions have you held at Princeton?

A. Well, I started as an lecturer, and then I was promoted to assistant professor, and then promoted to associate professor. I also served as the associate chair of the Department of Computer Science.

Q. Do you have tenure?

A. I do have tenure, yes.

Q. Now, what are your responsibilities as a professor at Princeton?

A. At Princeton generally I teach courses and I do research, perform research.

Q. What types of courses do you teach?

A. I teach compilers and computer architecture courses. These are courses for undergraduates and graduate students.

I also each an introductory computer science course for freshman. That course involves Java. We teach Java programming.

Q. You mentioned research as well. What types of research do you do at Princeton?

A. I direct a team of about a dozen graduate students, and the research is generally in the area of computer architecture and compilers.

Q. Do you have any industry experience, Dr. August?

A. Yes. While I was a graduated student I spent some time in industry.

In 1994 I went to Intel to work on microprocessor validation for one of their products.

In 1995 I went to Hewlett Packard here in the area to work on their Elcor and Trimaran compilers.

In 1996 I went back to Intel to work with them on a compiler that we had developed at the University of Illinois. I was helping them extend that compiler for their purposes.

In 1998 I worked with Lucent Technologies to extend the compiler we had developed at the University of Illinois, also for their purposes.

Q. Have you authored any technical papers related to your field of compilers and computer architecture?

A. Yes. I've authored over 80 papers in the area of computer architecture and compilers.

Q. Are you a member of any professional organizations or conferences?

A. Yeah. I'm a member of the IEEE and the ACM. These are the electrical engineering and computer science professional organizations.

Q. Do you have any patents, Dr. August?

A. I have one issued patent, yes. It's -- it relates to the optimization in a compiler to enhance performance.

MR. PAIGE: Your Honor, we would tender Professor August as an expert in the fields of computer science and electrical engineering.

MR. JACOBS: No objection.

THE COURT: Very well. If we stay in the area of his expertise, there will be no problem. Proceed.

BY MR. KAMBER:

Q. Professor August, were you retained by Google to proposed an independent opinion in this case?

A. Yes, I was.

Q. What were you asked to do?

A. I was asked to look at the accused devices and consider the issue of infringement or non-infringement of the '104 patent.

Q. Of the '104 patent?

A. Yes.

Q. Do you have any financial or personal interest in the outcome of this litigation?

A. No, I do not.

Q. Have you been at the trial for this second phase?

A. Yes, I have.

Q. Have you heard the testimony of Dr. Mitchell?

A. Yes, I did.

Q. And, in particular, did you hear his opinion that the Resolve.c functionality in Android infringes Claims 11, 39, 40, and 41 of the '104 Patent?

A. Yes, I heard that.

Q. And do you agree with that opinion?

A. No, I do not agree with that.

Q. Why not?

A. It's my opinion that the accused functionality does not infringe Claims 11, 39, 40, or 41.

Q. And just as a summary of the reasons, why don't you believe that the Resolve.c functionality infringes those claims?

A. Resolve.c operates on instructions that contain references. The '104 Patent requires that those references in the instructions be symbolic references, or symbolic references be in the instructions.

Resolve.c does not operate on instructions that contain symbolic references.

Q. Now, did you also hear Dr. Mitchell opine on the dexopt functionality and whether that infringes Claims twenty- -- excuse me, 27 and 29 of the '104 Patent?

A. Yeah. I heard that.

Q. Do you agree with his opinion regarding infringement of the -- of Claims 27 and 29 of the '104 Patent, by dexopt?

A. No, I do not. It's my opinion that dexopt does not infringe Claims 27 or 29 of the '104 Patent.

Q. Would you summarize for the jury why you hold that opinion?

A. Sure. For the same reason that the claims require that the instructions contain symbolic references. In dexopt -- dexopt does not operate on instructions that contain symbolic references.

There's an additional reason. That is that dexopt is a static process. And the claims and the Court's construction requires that the resolution occur dynamically rather than statically.

Q. Are you prepared to provide the basis for your opinions here today?

A. Yes.

Q. Okay. Before we do that, though, Dr. August, I understand that you've prepared a little bit of technical background for the jury; is that correct?

A. Yeah. I'd like to give a little bit of background so that I have some context to explain a little bit more about the patent and what's going on in Android.

MR. KAMBER: Let's pull up the slides, please. Thank you.

(Document displayed.)

BY MR. KAMBER:

Q. What do you want to provide by way of technology background, Dr. August?

A. Yes. So I would like to talk about data instructions in resolution. We've heard a lot about them, and I thought it would be useful just to review.

So I would like to start with data. And I've characterized data here into three categories:

Non-reference data. So that's data that is not a reference. Numeric references. Numeric references are data. They are reference data. And symbolic references. That's also data. It's a symbolic reference.

Q. Let's start with non-reference data. What is non-reference data Dr. August?

A. This is data that does not refer to other data.

In this example from the '104 Patent that we have on the screen, we see that there are two numbers here: 23 and 17. We know from the description of the patent that 23 and 17 are really just to be treated like integers, just numbers that we're all used to. They don't refer to any other data.

Q. Let's talk about numeric references as another form of data. What is a numeric reference?

A. So a numeric reference refers to other data by its location. It is also data that refers to other data by its location.

So here, in this example, also in Fig. 1A from the '104 Patent, we see that the 2 operand, the number 2 on the left side in the instruction stream, is referring to some other data in slot 2, the data object, by its actual location in memory.

Q. Let's talk about symbolic references, as well. The jury has heard a lot about symbolic references. What is a symbolic reference?

A. Well, it's not what's on the screen.

(Document displayed.)

THE WITNESS: There we go. So that is an example. On the screen now is an example of a symbolic reference. So a symbolic reference refers to other data by a name.

In this case, that name is not its location. So this "y" refers to some data "y" that's going to be labeled "y." But "y" does not give us any indication of the -- of the data's location.

BY MR. KAMBER:

Q. So what happens with symbolic references? How are they --

how do you find the data that they're pointing to?

A. Yeah, in the case of numeric references, when you have the location, you know where to go. You have the location.

In the case of symbolic references, we need to do a step called "resolution," where we take the name that refers to their data, and convert it to a location. So we may do something like a search.

In this case, we might search the data object for the data labeled "y." So that's what's being represented here, is after some resolution "y" is understood to be that second slot containing the "y = 17."

Q. Let's talk about instructions, and in particular, instructions bytecode, instructions in dex files.

But, just generally, what is an instruction in a virtual machine context?

A. Sure. An instruction is a small step that instructs the virtual machine what to do. An instruction may tell the virtual machine to fetch some data. It may tell the virtual machine to do some addition or some subtraction. It's a very small part of a larger program.

Q. Let's look at some instructions, but first you have a slide here. What do instructions in the context of a virtual machine, what are they made up of?

A. Yes. Instructions in general, not just for virtual machines, but instructions contain opcodes and operands.

An opcode describes what the -- the primary functionality of the instruction. So, for example, if you want to perform an addition, the opcode would be "add." If you want to perform a subtraction, the opcode would be "subtract." And made "get," if you want to get some data. That's the opcode.

Q. What is an operand?

A. So, the instruction also contains an operand. This is going to modify the operation.

So, in the case of the "add," what do we want to add? In the case of the "get," what do we want to get? Where do we want to get it from? It helps the -- it completes the concept of the instruction as a step.

Q. And are there instructions, looking again at Fig. 1A of the '104 Patent, are there instructions shown here in this figure?

A. Yeah. Actually, with that tag 14 we see a "load 2" instruction, in the left side labeled "instruction sequence."

Q. Is that "load 2," is that an instruction in the sense of having an opcode and an operand?

A. Yes. Here the instruction "load 2" has both an opcode, "load," and a single operand, "2." So this will show what I just said in a little more detail.

Q. The load -- well, what is the "load" here, Dr. August?

A. So this is the opcode. It's the basic operation. And these are the examples I gave earlier.

Q. What is the "2"?

A. In this case, the 2 is an operand. In this case, in the case of the load, the 2 indicates where to -- from where to load the data. In this case, it's a location, a numeric reference.

Q. Let's look at some examples of instructions.

What is on the left of this screen? This is slide 14 of the demonstratives.

A. On the left side we see an instruction sequence. The instructions here have been replaced by dots, just to indicate there are instructions in those slots.

Q. Let's take a look at an example instruction.

What is happening here with the -- what does "Load R1,3" stand for?

A. So this is similar to the load instruction we saw earlier. This load instruction is appropriate for a register-based architecture, rather than a stack-based architecture. The "R1" being a register. It's just another place we can put data when it's been fetched or loaded.

This says goes to memory location 3. I think we can advance this. Says go to memory location 3 or slot 3. The system will go there and grab the data, "31" and load that data or store that data in register R1.

Q. Let's take another example of an instruction, this time containing a symbolic reference. What happens when an instruction contains a symbolic reference?

A. So, again, the load instruction tells the virtual machine to get some data. In this case, though, we don't know where that data is. We know the data is called or named "y." So we need to figure out how to find that data.

So we'll perform a step called resolution. We'll take that "y" and we'll perform the resolution process. It may involve a search, again.

So here we might go to the first slot and understand that the first slot does not contain data labeled "y." And we'll go to the second slot, and we'll see here that the second slot contains the data labeled "y." So we go to that location, slot 2, having resolved the symbolic reference, and grab the data, in this case "17," and put it in the register R1 specified by the first operand.

Q. Let's take a look at another instruction. This is called, "Add R1,2,2." How does this instruction work?

A. Yeah, so this is an instruction that doesn't contain a reference of any kind. Here, the "2" is what's called immediate data. It's data that's in the instruction.

And we take -- the operation of this instruction would involve taking the first -- I'm sorry, the second and the third operand, the "2" and the "2," adding them together. 2 plus 2 we get 4.

Q. Do that again, Dr. August. Excuse me.

A. Go a little slower.

We take the operands, the 2. Treat that as a number, not as a reference. We take the last operand, the third operand. That's also a 2. We'll treat that as a number. We'll add the two numbers together, 2 plus 2. We'll get 4. And then as the instruction indicates, we're going to put that result into register R1. And that's what we've done here.

Q. Now, what are computer programs -- what constitutes a computer program?

A. So a computer program contains instruction sequences. And, also, a computer program contains data.

Q. How do instructions relate to computer programs?

A. So an instruction -- a computer program is going to be composed of instructions and data. But one instruction typically doesn't make an entire program.

A program is going to contain many tens, hundreds, thousands of instructions because, generally, you want to do more than just add. You want to play Angry Birds, for example. You have to do a lot of math to do that, for example.

Q. Let's talk about resolution for a moment. You referred to it a little bit in this tutorial.

What is resolution, generally speaking?

A. So, as I mentioned earlier, resolution is the process of converting symbolic reference to numeric memory locations. As I described earlier, it may involve a search or other process to find the numeric memory location labeled or named by the symbolic reference.

Q. And what is this slide showing? This is slide 23 from the demonstrative, Dr. August. Do you have an illustration here of resolution?

A. Yes. So we're going to resolve the "y" in "load y." Remember "load y" is going to get the data labeled "y" and fetch it.

In this case, what we want to do is search the data object. So we'll go to the first slot and ask ourselves -- this computer will ask the question of itself. It will say, is slot 1 labeled "y"? And the answer is no. The search will continue in slot 2. The process continues.

In this case, it realizes it has found data labeled "y," and will succeed in the search, and understands that now the data is in memory location 2, slot 2. And it will perform the operation, in this case a fetch or a load.

THE COURT: Can I ask you a question?

THE WITNESS: Sure.

THE COURT: What if it was the very last one?

THE WITNESS: Then it would take --

THE COURT: Would take a lot longer to get there.

THE WITNESS: Yeah, that's actually -- that's actually the problem with symbolic references, is that they take a long time, sometimes, especially -- as you observe, we're doing what's called a linear search from the beginning to the end.

If we need to do that linear search and the data we want is at the end, it will take longer than if the data is at the beginning. So we try to avoid resolution for that reason. It is time consuming. It's generally not just a single instruction.

In the case of a load with a numeric reference, we can perform the load with a single instruction. With resolution, we may have to perform many steps.

THE COURT: You just have five in this example, five positions.

THE WITNESS: That's right.

THE COURT: But in real life, would there be 5 or 5,000 or 5 million? Give us a rough idea how many would be -- you'd have to search.

THE WITNESS: It could be -- it could be anywhere from one to millions. It really depends on what the program is doing and how much data we have.

If we have a little bit of data, then it's easy to do. Or a -- or a few locations, names for locations. But as we increase the names, we give more and more names to data, we're going to have to have more searching. So it could be thousands, maybe tens of thousands in some cases.

In some sense it is limited -- in some ways in which symbolic references are used, it's limited by the programmer and the program. In this case, the programmer had to type "y." So the programmer will only make a certain number of symbolic references.

THE COURT: Is it just one column like that, or could you have a column that had all the Ys? You'd have y1, y2, y3. So you'd have a column for the Ys, but another column for the Zs. So that at least you wouldn't have to search the Zs.

THE WITNESS: Yes. What you're getting at is, there's an optimization you can do to improve the performance of resolution. You can organize the data. As you're suggesting, it's actually in a sorted manner.

If you sort your symbolic references, the data for your symbolic references, then you can find the symbolic references faster because you know that in your example you could go to the "y" column to see all the y1, y2, all the references that begin with "y." So that that's actually, what Your Honor suggested is one of the optimizations performed.

THE COURT: Say that again.

THE WITNESS: I'm saying that you're a good computer scientist.

(Laughter)

THE COURT: Well, thank you.

(Laughter)

MR. VAN NEST: We're hiring down at Google, Your

Honor.

(Laughter)

THE COURT: All right.

MR. VAN NEST: In case you get tired of this job.

THE COURT: All right. Go ahead.

MR. KAMBER: Thank you, Your Honor.

BY MR. KAMBER:

Q. Can you please give an everyday example of -- I don't know if I asked this already, an everyday example of resolution?

A. Yeah. So Mr. Van Nest, in his opening, had a great example that I like. It was The White House.

So resolution of the -- the symbolic reference of "The White House" may involve going to Google Maps, typing in "The White House" and having it return to you "1600 Pennsylvania Avenue." That would be a real-life example.

Q. Let's change gears a little bit and talk about the '104 Patent that's asserted by Oracle in this case.

What is the '104 Patent about, generally?

A. The '104 Patent describes a way of executing instructions that contain symbolic references.

Q. And how does the '104 Patent relate to a symbolic reference specifically?

A. Uhm, the '104 Patent describes how to more efficiently execute instructions that contain symbolic references.

So, we have some examples in the patent that may help illustrate the resolution or the way in which the '104 Patent handles symbolic references and instructions.

Q. What do -- well, we have here slide 26, showing Fig. 8 of the patent. Do you have that in front of you, Dr. August?

A. Yes.

Q. What does Fig. 8 show regarding the '104 Patent?

A. So it's showing how to handle a symbolic reference inside of a load instruction, and how to improve the performance of executing that load instruction with a symbolic reference.

The steps here shown in Fig. 8 involve resolving the reference, symbolic reference "y," finding its location, in this case slot 2, and remembering its location by overwriting the instruction "load y" with an instruction "load 2."

Q. Where, if anywhere, does the '104 Patent require that the -- that the symbolic references be?

A. Well, the claims require that the symbolic references be inside the instructions.

Q. I believe that we have -- on slide 27, this Claim 11 is reproduced from the patent. Do you see that, Dr. August?

A. Yeah. This is a representative claim. It says here, highlighted, "Instructions containing one or more symbolic references."

So the apparatus described here is going to operate or have instructions containing one or more symbolic references.

Q. Is that requirement also reflected -- to what extent is that requirement also reflected in Fig. 8, for example?

A. Well, we see it throughout the patent. It's in Fig. 8. It's in the claims. It's also in the description of the figures.

Here we have -- I've highlighted one example of that. It says here, "load instruction." That "14" is just referring to the figure, the load in the figure. It's initially generated with a symbolic reference.

If you generate a load with a symbolic reference, that load instruction is going to contain that symbolic reference.

Q. So given a little background, talked about the '104 Patent. Let's talk about Android and what's accused in this case.

Are you familiar with the Android code accused of infringing by Oracle?

A. Yeah, I'm very familiar with that code.

Q. How much time have you spent reviewing the code for Android?

A. Uhm, in relation to this case, more than 50 hours.

Q. And are you familiar with the term "dex file"?

A. Yes.

Q. Are you familiar with the format of dex files?

A. Yes.

Q. Okay. What is a dex file?

A. A dex file is a Dalvik Virtual Machine program file. It contains both instructions and data that instruct the program on how to perform an operation like Angry Birds or something like that.

Q. Do dex files contain symbolic references?

A. Dex files contain symbolic references, yes.

Q. Do the instructions -- does the instruction stream in a dex file contain symbolic references?

A. When -- when you find symbolic references in a dex file, they will not be contained inside instructions. You will find plenty of symbolic references outside of the instructions, but as you go through the instructions you will not find a symbolic reference. Not a single one.

Q. Were you here in the courtroom during Mr. McFadden's testimony?

A. Yes, I was.

Q. Have you reviewed the demonstrative exhibit that he had prepared to show the process of resolution in Android?

A. Yes, I reviewed that figure.

Q. And based on -- and you said you've reviewed the code, correct?

A. Oh, yeah, I reviewed the code and the figure.

Q. To what extent do you agree with Mr. McFadden's summary or explanation of how Resolve.c and dexopt work?

A. I think that figure and Mr. McFadden did a great job of explaining how the code works at a very -- at a high level. It's entirely accurate. There's some abstraction, of course, because we don't want to look through all the code. We want to understand the code.

Q. So let's talk about your analysis of infringement issue for a moment.

Did you use any definitions from the Court in performing our analysis?

A. Yeah. I looked at the Court's claim construction order and also the construction of a term.

Q. Let's go back. Let's go back. This is slide 29. There's a definition here of symbolic reference. Could you please read that.

A. Sure.

> "A symbolic reference is a reference that identifies data by a name other than the numeric memory location of the data, and that is resolved dynamically rather than statically."

Q. And let's start by talking about Resolve.c. Why is it your opinion that Resolve.c does not infringe the '104 Patent?

A. '104 Patent and the Dalvik Virtual Machine does not operate on instructions that contain symbolic references.

Q. Let's take a look at the demonstrative from Mr. McFadden.

And we're looking here -- well, what are we looking at here, Dr. August?

A. Well, this is the demonstrative you were referring to earlier, from Mr. McFadden.

Q. Where is the instruction stream on this example?

A. The instruction stream is under the -- on the left side, in the light blue. Everything on that side in the light blue is an instruction. There are no instructions to the right of that.

Q. Pardon me.

Where are the symbolic references in this figure?

A. So we see some symbolic references here in the string data table, for example, "fun" was the example that was covered most heavily during his testimony.

Q. To what extent does the "1" index in the instruction stream represent that symbol "fun" in the string data?

A. Well, it doesn't represent the symbol "fun." It's a -- it's a location of data in the field ID table.

In particular, that operand, "01," indicates that -- as we follow the arrow, indicates that in the field ID table we'll find other data.

Q. And what does it find once it reaches that entry marked "01" in the field ID table?

A. So in this case, the other data is two pieces. There's a "02," which is also -- which is a numeric reference, a location of more data, and a "76," which is, again, the location of more data.

So we have in this case two places to go from the field ID table.

Q. Are those known as indexes, Dr. August?

A. Yeah. Actually, the "01" in that instruction is an index into the field ID table. An index is another way of saying location in a table. Memory location.

In the field ID table, we see the "02" and the "76." These are also indices. "02" is an index into the string ID table, and "76" is also -- "76" is also a location in the string ID table.

THE COURT: We're going to need to take our next break soon, but I want you to finish or come to a good resting point.

MR. KAMBER: Let me just ask two or three more questions here, Your Honor.

THE COURT: All right.

MR. KAMBER: Thank you.

BY MR. KAMBER:

Q. So let's just trace one of those through.

The index "02" from the field ID table, to the entry, the slot 02 in the string ID table, what -- what is the data that's there?

A. Yeah, so here the "02" in the field ID table is referring, again, to more data. In this case, the data is another index in the string ID table.

And anticipating where you're going with this, I think we're going to talk about the index in -- the data, the "08," is an index into the string data table as we follow that arrow.

So starting with the instruction, we have "location." Going to the field ID table, we have two locations. We'll follow the yellow location to another location in the string ID table; and then, finally, at least on this slide, to a location in the string data table, where --

Q. Did -- excuse me.

A. Go ahead.

Q. Did you hear Mr. McFadden testify that that "08" in that string ID table is an offset?

A. Yes.

Q. And I believe you just referred to it as an index. What is the difference or -- if any, between an index and an offset?

A. So an index and an offset are often used interchangeably. They are both numeric memory locations.

Because we are short on words we may assign, in different teams or in different contexts, slightly different meanings.

So in the Dalvik Virtual Machine, we might use an offset to mean a numeric memory location from the beginning of the dex file; whereas, we may use the word "index" to refer to a location in a constant pool.

This is a convenience that programmers often use. They can add language or meaning to the language.

Q. "Indexes" and "offsets" both refer to locations in memory?

A. They are both numeric memory locations.

MR. KAMBER: This is a good place to stop, Your Honor.

THE COURT: All right. Before we break, I want to leave a question with the witness and with counsel. And when we come back, maybe you'll help us. But let me ask a preliminary question.

Do you understand what the difference is between you and Dr. Mitchell over this symbolic thing?

THE WITNESS: I believe I do.

THE COURT: When we come back, can you explain it using that diagram on the screen, the difference? In other words, we keep hearing your view, but is there a way to tee up the issue so we can all understand it better?

THE WITNESS: I would be happy to do that.

THE COURT: Do you think you can do that?

All right. That's where I would like to start when we resume. Okay. Fifteen minutes. Please remember the admonition.

THE CLERK: All rise.

(Jury out at 11:19 a.m.)

THE COURT: Please be seated. Are there any issues for the judge?

Remember, now, we're going to have to break at 12:15. I'm sorry, I have to leave a little early today.

MR. VAN NEST: When we break, you're going to break and go?

THE COURT: I'm going to break and go. I might give you three or four minutes, but that's about it.

MR. VAN NEST: Okay. Fair enough.

THE COURT: Do you need more than that? We will have to stop the witnesses earlier then.

MR. VAN NEST: Okay. I don't think we do, but I just want to figure out what the protocol would be. That's fine.

THE COURT: How much longer on direct with the witness?

MR. KAMBER: Probably about 20, 25 minutes, Your Honor.

THE COURT: Okay. Great is.

MR. VAN NEST: Your Honor, we do have a couple of exhibits it would be good to move in, if we could.

THE COURT: Yes.

MR. PAIGE: Just housekeeping. These are the deposition excerpts played by Dr. Mitchell yesterday.

THE COURT: Fine. What is it?

MR. PAIGE: 3541. And then 3542 are Mr. Gupta's today.

(Trial Exhibits 3541 and 3542 for identification.)

THE COURT: Is that it?

MR. PAIGE: Yes, Your Honor.

THE COURT: Great. See you in 15 minutes.

(Recess taken from 11:20 to 11:31 a.m.)

THE COURT: Okay. Let's go to work.

MR. VAN NEST: Your Honor, before we start, I think we have an agreement on that one transcript change. Do we not?

MR. PETERS: We do. Do you have the citation?

MR. VAN NEST: Yes. Your Honor, at page -- in the record transcript at page 3463, line 19, we're in agreement that the words "HTC Droid Incredible"' should be stricken from the transcript.

MR. PETERS: That's right, Your Honor.

THE COURT: I wonder how we do that. Let me go offline for a minute and ask my court reporter.

(Discussion off the record.)

THE COURT: All right. The court reporter will be directed, before she files the original, to strike those words from that page, since she has informed us she has not yet filed the original.

So that will be the order of the Court.

MR. VAN NEST: And I don't think this is significant enough for you to interrupt and tell the jury anything about it. Are you going to list the products in the jury instructions?

THE COURT: I don't think so. We had that discussion yesterday. You weren't here for that.

MR. VAN NEST: Right.

THE COURT: You must have been at the tennis court.

MR. VAN NEST: I'll take that as a joke.

(Laughter)

MR. VAN NEST: But the "HTC Droid Incredible" will be stricken from the record. And I guess -- I guess there's no -- I guess there's no reason to instruct the jury. I think it was so insignificant.

THE COURT: I agree. Okay. Can we get started?

MR. JACOBS: Sorry, Your Honor. Yes.

(Jury enters at 11:36 a.m.)

THE COURT: Welcome back. Please be seated.

We only have about maybe 30, 30 more minutes to go, 35 more minutes to go this week. You all set with your notepads ready? Great.

Counsel, go ahead.

BY MR. KAMBER:

Q. So before the break, Dr. August, the judge asked a question about what the difference was between you and Dr. Mitchell regarding this figure that is up on the screen for the jury.

Could you please explain where you see the differences in opinion.

A. Yes. As I understand it, Dr. Mitchell is trying to get the symbolic reference from the string data table into the instruction. I think there are two possible ways that that's being attempted.

The first is that somehow, after going to the instruction, following a numeric reference, a numeric reference, a numeric reference to a symbolic reference, that that symbolic reference changes or -- the quality of the references that led to that symbolic reference, making the numeric reference in the instruction a symbolic reference.

THE COURT: Wait. Say that again, more slowly.

(Laughter)

THE COURT: You can draw like John Madden.

THE WITNESS: Oh.

THE COURT: See if it will work.

THE WITNESS: Oh, okay.

THE COURT: Do a John Madden and show where the rubber meets the road on this issue.

(Witness marks on electronic slide.)

THE WITNESS: Okay. So if the instruction has a numeric reference, the numeric reference is followed. And we find a numeric reference. We're okay. The instruction contains a numeric reference.

We follow another numeric reference to another table. We're still okay. The instruction still contains a numeric reference.

We follow a numeric reference to a symbolic reference. Now, the original numeric reference in the instruction is somehow symbolic because of its, I guess, relationship or having been infected by this symbolic reference all the way on the end here. That's the first way that I think --

THE COURT: The premise being that the certain of said instructions containing one or more symbolic references. And you're saying that -- well, what is "01"? Is that, in your view, a symbolic reference?

THE WITNESS: No. "01" is a numeric reference because it gives you an actual numeric memory location. Right here (indicating). This is the location, 01.

There's no resolution, no search. Nothing -- nothing expensive about figuring out what that instruction is referring to when it goes to the field ID table.

THE COURT: You said there was a second possible issue.

THE WITNESS: Yeah. So the second -- I think the other way to do it, you could bring the symbolic reference into the instruction. So kind of this "fun" at the end is -- is changing the character of that numeric reference in the instruction.

Another way to -- to deal with this problem of instructions containing symbolic references would be to say that the instruction is not just this (indicating), but the instruction is all of that (indicating). And that would bring the symbolic reference into the instruction.

MR. KAMBER: Can I just ask a clarifying question, Your Honor?

THE COURT: Go ahead.

MR. KAMBER: It may be a little leading, but I want to be sure.

BY MR. KAMBER:

Q. You said there's two differences, correct?

A. Two differences.

Q. There are two ways to characterize what you think?

A. Two possible ways.

Q. And the first was that the symbolic reference infects all of the numeric references before that, correct?

A. That's right.

Q. And the other way is to say that you're drawing this box around all of the instructions and the numeric references that go in between, and that's another way --

A. Do you want me to tell you what I think about these?

Q. Yes, please.

A. Okay. Well, I think symbolic references have been in computer programs since before the '104, so you would have the symbolic references all over the place.

The symbolic reference in this case exists in the data. To say that we're going to treat or understand this -- this numeric reference in the instruction as a symbolic reference, even though it tells us a numeric memory location, is a stretch, in my opinion.

THE COURT: What would be -- where you have the "01" there, give us an example of what would be a symbolic reference, in your opinion.

THE WITNESS: Oh, okay.

Well, if you took the letters f-u-n and you actually put them -- can I erase this somehow?

THE COURT: There is a way to.

MR. VAN NEST: Clear.

THE COURT: There is a way to clear. Dawn, how do we do it?

(Screen markings cleared.)

THE COURT: There we go.

THE WITNESS: Yeah, so if you actually took f-u-n -- and this f-u-n is taking up three positions. It's just compressed here horizontally. But if you actually did f- --

THE COURT: I don't like that f-u-n thing. Use "y" or "x." Use "y" or "x."

THE WITNESS: Yeah, so if you have a symbolic reference, let's say "y" or "x," and instead of "01" you put a "y" or an "x" in the instruction, then of -- then you would have a symbolic reference --

THE COURT: What would "x1" be?

THE WITNESS: Well, then you would have "x" and a "1."

THE COURT: No, no. I mean "x1" run together. Is that a symbolic reference?

THE WITNESS: "x1" -- "x1," if it's a name for other data that's not its location then, yes, it would be a symbolic reference.

THE COURT: All right. So are you saying that in the Android instructions, you've looked at them and you never find an "x" or a "y" or -- it's always a number?

THE WITNESS: I've looked at every single implementation of the instructions, and I can say with certainty that there is not a symbolic reference in the instructions.

You'll never see "y" or "x" or "z" referring to data by a name other than a memory location, in the instructions themselves.

THE COURT: Is that in controversy between the two experts, or is that -- is that common ground?

THE WITNESS: Uhm, I --

THE COURT: Do you know for sure?

THE WITNESS: I don't know for sure. But the -- I think there is understanding on part of both of the experts that you do not see in the instruction stream the characters "y," "y1," "y2," and so on.

I think that's pretty clear in the -- in the part of the code called "insns," or instructions, that lays out the instructions and all the documents that you've seen so far that describe the formats of the instructions, you never see a place in that sequence of bytes, in that sequence of instructions, to hold a name like "y" or "x" or "y," "x1."

THE COURT: All right go ahead, Counsel.

BY MR. KAMBER:

Q. With respect to the difference, it's -- Dr. Mitchell is calling the index in that "01" -- perhaps we can clear the screen -- the "01" in the instruction stream that index a symbolic reference, correct?

A. Yes. He's calling that index -- that's -- that's -- to get the symbolic reference into the instruction, you have to call the index here a symbolic reference.

Q. Why do you disagree with Dr. Mitchell that an index -- well, let me rephrase. Why do you think that an index is a -- is not a symbolic reference?

A. There's -- there's no resolution. There's no question that "01" is referring to a memory location. The memory location referred to by "01" is this position over here, "01."

I should point out that this "01" is actually meant to illustrate the location. This is -- the "01" -- let me clear this. Excuse me.

This "01" right here, that's not data. That's just the -- a representation of the position of "02" and "76." So this position "01" does not exist over here (indicating). It's just we know that "01" is the second -- because we count from zero -- the second position, the second entry, the second slot in the field ID table.

"01" in the instruction says the location in the field ID table is the second slot because we started counting from zero.

Q. To what extent does that "01" tell you to go to a particular location in the field table?

A. It's the actual memory location.

"01" is an index that specifies a position within the table, within memory, of the data that this "01" is referring to.

The operand in the instruction, we've seen that before as field@CCCC. That "CCCC" in this case, is the "01." Here we have "field@." The "@" sign indicates that we're talking about a location. You're at a location. So here it's at location "01."

THE COURT: Can I ask another question?

We've all seen from the way the programs are written that they write them with variables like "x" and "y" in the actual Java code. They wouldn't write it with -- they wouldn't write it with the position in memory. They would say "x" or "y."

So if you're right, then at some point before these instructions are formed, that "x" or "y" has to get changed into that "01." So where does that step occur?

THE WITNESS: So in this case, the step occurs in javac.

Javac will see in this case a "y" in the programmer's code. Something like "y = 1." And javac will see, you know, some statement with a "y" in it.

And javac understands that to create instructions, it needs to remove -- it needs to create instructions without symbolic references. It moves the symbolic references into the string data table. And the instructions will now, through various levels of indirection, refer to that string. But the instructions are, at that point, symbolic-reference free.

THE COURT: What is javac?

THE WITNESS: That's the Java compiler.

THE COURT: Is that a step before this or after -- where does that step occur on the screen?

THE WITNESS: Do you want me to draw the -- okay.

THE COURT: Has that already occurred over here?

MR. KAMBER: No, no. I was going to allow the witness to draw.

THE COURT: Please, go ahead.

MR. KAMBER: Just a moment, Your Honor.

THE COURT: That's good. Go ahead. Give us a --

MR. KAMBER: Hold on just a second. Going to get you a bigger piece of paper.

(Witness drawing at the easel.)

THE WITNESS: Okay. So let's start --

THE COURT: Keep your voice up now. The jury has got to hear you.

THE WITNESS: Let's start all the way back at the beginning.

And we know that that's a Java program. So let's -- I don't want to use "fun."

(Laughter)

THE COURT: We don't like that word.

THE WITNESS: I'll say "y."

So this is Java. "Y" equals "Hello World," for example. You're familiar with the HelloWorld program.

"Y" is a -- perhaps it's a field name. And in this Java program, we're taking "Hello World" and we're going to store it in the location referred to by the symbolic reference "y."

The next step is the execution of the Java compiler.

THE COURT: Is that what you meant by "javac"?

THE WITNESS: Javac. Yeah, Java compiler. The way it's -- it's invoked on a computer is you type "javac" and the name of this file. And it will compile the file.

Then that will produce a class file, okay. And in that class file we have both instructions and data. And at this point we have some instructions -- I'm going to use the same numbers we had in the example here, but these are actually going to be -- this is a Java class file, not a dex file. So it's -- there's going to be a different kind of bytecode.

But just to keep things consistent with the figure on the screen, I'm just going to use the instruction "52," "01." And "01" is the numeric reference. And in this case the "y" has been placed over here.

This operand takes us to a table, constant pool table. And in that constant pool table we're going to find another numeric reference. I'm not going to go through all the steps, but it looks like that (indicating). So that's the class file.

I can continue, if you would like me to continue this out to show how this ultimately ends up as a dex file, if you would like me to do that, or I could stop here.

THE COURT: Is what's on the screen the dex version or --

THE WITNESS: Yeah. So let me take it to the dex file so we're actually talking about the same thing.

The next step is another compiler called "dx tool." And dx tool is going to take the Java class file and it's going to produce a dex file. Now we're getting much closer to that (indicating).

So that dex file also has instructions and data. The dx tool does a number of things. But one of them is not to remove or place the symbolic reference in the instructions. It's not going to take the symbolic reference out of the data portion. So we will still find the symbolic reference in the data, in the dex file. And now -- "52," "01." Now we're there (indicating).

Okay. In the instructions in the dex file, we have numeric references, references to locations, memory locations. And those references have other references to other data, other references, and, ultimately, to a symbolic reference.

And I could show, also -- there's another step, once the dex file gets installed on the device. I can continue, if you would like me to do that. Or, perhaps, we could do it later in my testimony.

THE COURT: No, that -- which one of those steps -- there are two sets of instructions. There's the Java set and then there's the dx set, dex set.

In terms of the patent and what you're fighting over here, is it the dx set you're fighting over, or the other set?

THE WITNESS: It's this one (indicating).

THE COURT: Now, go back to my original question.

I see you put the "y" there. All right. That's your view, the "y" goes there. But where does "Hello World" go? Where did "Hello World" go on your diagram?

THE WITNESS: Yeah, so "Hello World" would also go into the data section. "Hello World" is just a couple of words. It's not an instruction saying do an addition or a subtraction or a fetch.

So "Hello World" would also be in this data portion of the program. So I'm going to write it down here. But it would actually -- it would actually be in the same strings table as the symbolic reference.

So you would see -- maybe I won't write "Hello World," but it would exist there as Hello World (indicating), under -- adjacent to perhaps the "y" for the field reference spot.

THE COURT: Where would it be in the dx one?

THE WITNESS: Similar location (indicating).

THE COURT: So it sounds to me like a key point that you make on where you disagree -- and I'm not saying who's right or wrong, believe me. I'm just trying to find out what the issue is.

You're telling us that over there on that bottom box, that the instructions never contain an "x" or a "y," they always contain a number?

THE WITNESS: They always contain a number that refers to a memory location. They will never contain the actual what we call the string, which is just the sequence of characters "y" or h-e-l-l-o. They'll never contain that sequence of characters. You will not find that in the instructions.

I don't think there's any disagreement about that.

THE COURT: So, again, what is the disagreement then?

(Laughter)

BY MR. KAMBER:

Q. What is the disagreement?

(Laughter)

A. I think somehow this -- this "y" has to be -- this symbolic reference in the data portion has to be in the instructions to meet the claim language. The claims say, literally, that the symbolic reference is in the instruction. So somehow "y" has to be in the instruction (indicating).

So maybe it's some kind of transitive property, where you go numeric reference, numeric reference, numeric reference, symbolic reference. But the symbolic reference is not an instruction.

Or maybe -- maybe the other possibility is the instruction isn't the instruction, what we know as instructions, with operands and opcodes. Maybe the instruction is more than that.

Maybe the instruction is, you know, some -- some slice of the entire, you know, program through several tables to finally include within it the symbolic reference.

That, I think, is not a reasonable interpretation of the meaning of "instruction" from a computer scientific point of view.

THE COURT: All right. Unless you need him to stay there, I want you to be able to end the week on -- with whatever you wanted to do. And I know I have used up some of your time.

MR. KAMBER: That's fine, Your Honor.

THE COURT: I'll stop asking questions.

MR. KAMBER: They are all good questions, Your Honor.

I'm going to have the witness stay down there just a minute because I want to draw one thing before we end the week.

But let me ask about three issues.

BY MR. KAMBER:

Q. Dr. Mitchell says that an index is a symbolic reference, correct?

A. Sometimes.

Q. And indexes are what's in the instruction stream, correct?

A. Yes.

Q. Now, is there a disagreement between you and Dr. Mitchell as to whether or not the instruction stream contains indexes?

A. No, there's no disagreement. The instructions contain indexes (indicating).

Q. Now, why would you -- why would you move the "y" out of the instruction stream and put it into string data or the string table?

A. Sure. And this is -- it's -- it's about space in the instructions, about efficiency of executing the instructions. I think Mr. McFadden discussed this earlier.

You want to -- if you have an instruction like "52," like "get," like "load," you would like it to be a fixed size. You would like it to be 52 and an index. An index is a fixed size. It's a number. "CCCC" is four digits, let's say. Actually, it's hexadigits. And you know that the -- that it will always be four digits.

If you have a symbolic reference in the instruction, like "y" or -- I apologize Your Honor -- "fun," it's a different length.

So, now, interpreting these instructions in the virtual machine is much more difficult because you have to first get the opcode. Then you have to figure out how long the symbolic reference is in the instruction. Is it one character, like "y"? Or is it three, like "fun"? Or is it many more, like "Hello World"?

Once you have figured out how long the symbolic reference is, then you have to go and get the symbolic reference out of the instruction. Find h-e-l. Know when to stop. Because if you don't stop, you're going to start interfering with the other instructions or misinterpret the other instructions.

So all that is much less efficient than just knowing that an instruction that gets data is, in this case, as is represented on the screen, just two numbers: 52, 01. Regardless of the length of the identifier, the variable, or whatever you want to call the field.

Q. Now, there was also a question from the judge regarding, is the dex file the issue that we're talking about? Is that the -- do you remember that?

A. Yes.

Q. Is there also another version of this file that's at issue in the infringement allegations from Oracle?

A. Yes. So the dex file is what's installed on the device.

You go to the app store. You click on "purchase." And the dex, this file is what's delivered (indicating). This comes onto the device.

So this line I'm drawing here, this is somewhere else. This is your phone. (Witness indicating.)

When it arrives on your phone --

Q. Can I just interrupt --

A. Sure.

Q. -- and have you sort of just write "phone" at the bottom.

A. Okay. Sure. (Witness complies.)

Q. Thank you.

What's above there? How would you characterize that?

A. Well, it's the app store, the developer's computer, the programmer's, you know, source files. These are the computer that the developer is using, the person writing the program --

Q. Can we just say "developer"?

A. I'll just write "D-E-V." How's that?

Q. Sorry, I interrupted you.

A. Sure. So when the program that's been purchased arrives on the phone, the -- what it's going to do is go into dexopt. That's going to process this file, and produce another file called an odex file, which you've heard of. Just say opt, optimized dex file.

And that optimized dex file looks very much like the dex file -- except it has some other things in it, like quicken instructions that you've also heard about -- but its main structure is intact. It has both instructions and data. The instructions still contain only, as far as references are concerned, numeric references, locations in memory. And the symbolic references are still in the data portion.

As Mr. McFadden went through, perhaps this "52" is replaced with a "137," or whatever it was, as the code got optimized by the dexopt tool.

At this point, this file, there's been no execution going on. We haven't -- the owner of the phone has not started the application. But, at some point, perhaps maybe weeks later, maybe immediately, the application will get started. And this optimize dex file will get processed by the virtual machine.

And, actually, in particular, it's the interpreter. I'm just going to write "interp" to refer to the interpreter.

And that interpreter is going to go through the instructions, as necessary, dynamically processing those instructions to perform the operation described by the programmer all the way back here (indicating).

MR. KAMBER: Two quick questions, Your Honor, then I think we can finish.

THE COURT: All right.

BY MR. KAMBER:

Q. First, where is the Resolve.c functionality on that drawing?

A. So Resolve.c is here (indicating), in the interpreter.

Q. Is there a dividing line here between runtime and -- well, between dynamic optimizations, dynamic linking and static linking?

A. Sure. So, here the application arrives on the phone. We do some install time work. That's a static process. It happens once. It happens before you start the application. It happens before there's any dynamic execution of the application.

That produces this dex -- odex file, optimized dex file. That's a file that's placed on the file system in the device. And at some point later, when the user clicks on the icon, this file that's been stored will then get executed.

So somewhere like this, we have a difference between static and dynamic. So it's a -- write this like this. Okay.

And the interpreter executes the program. The user can use the program. That's a dynamic behavior. That's the execution of the program.

THE COURT: One more question, or are we ready?

MR. KAMBER: No, that's probably a good stopping point, Your Honor.

THE COURT: All right. You can come back up here. We're going to wind up for today.

I think we're on track to finish the evidence on Monday morning. We might even have the closings. I'm hoping we will have the closings on Monday morning.

You need to start -- for certain, the case will be to you by Tuesday. I think it will be to you possibly Monday.

And you need to, as before, start thinking about days you might want to work in the afternoons on the case. You don't have to commit to any of that right now. But you might want to be thinking ahead for that part of the schedule.

So, remember the admonition. We're going into a weekend. You'll be walking along and somebody will show you a browser, and you might be tempted to put something in there that you shouldn't. No. Please don't do that. Don't talk to anyone about the issues in the case. Don't do any homework. Keep an open mind. All of the things that I told you before.

We so much appreciate the hard work you are putting into this. You are making the most tremendous sacrifice for your country. Almost no jury has ever served as long as you. Very few.

(Laughter)

THE COURT: And we very much appreciate it. And we'll see you back here.

The lawyers, I want you to know, are working hard to find ways to shorten up the case because they also appreciate how hard you've been working.

See you back here on Monday morning. Thank you.

(Jury out at 12:08 p.m.)

THE COURT: Okay. Be seated, everyone.

The witness can step down. We will need you back here on Monday, too. I hope you enjoy yourself. Are you going to stay over the weekend?

THE WITNESS: Yes.

THE COURT: Good. I hope you have a good time here in San Francisco.

Any issues for the Court? I have a few minutes before I've got to run.

MR. KAMBER: The only thing I wanted to do was mark this drawing as TX 3544.

THE COURT: Fine.

(Trial Exhibit 3544 marked for identification.)

MR. JACOBS: Nothing from us, Your Honor.

THE COURT: All right. Remember, you need to have it in writing what issues you want to go to the jury in this phase, and sign off on it. I just can't mediate that between you. It's got to be full agreement. So whichever way you want to jump on that is okay with me.

Let me see how much time we used. I can't -- I think you both have -- you both must be keeping track. If you insist, I'll try to figure out your time. But I don't -- it will take me five minutes to add it up.

MR. VAN NEST: Our legal assistants have been doing that, Your Honor. They can figure it out. If we have a problem, we'll let you know.

THE COURT: So, have a good weekend. See you back here on Monday at 7:30.

MR. VAN NEST: Thank you, Your Honor.

MR. PAIGE: Thank you, Your Honor.

MR. JACOBS: Thank you, Your Honor.

(At 12:09 p.m. the proceedings were adjourned until Monday, May 14, 2012, at 7:30 a.m.)

I N D E X

DEFENDANT'S WITNESSES	PAGE	VOL.
MCFADDEN, ANDY		
(PREVIOUSLY SWORN)	3729	21
Direct Examination Resumed by Mr. Kamber	3729	21
Cross Examination by Mr. Jacobs	3747	21
Redirect Examination by Mr. Kamber	3762	21
Recross Examination by Mr. Jacobs	3767	21
Further Redirect Examination by Mr. Kamber	3770	21
GUPTA, VINEET		
Video Deposition played	3771	21
PARR, TERENCE		
(SWORN)	3772	21
Direct Examination by Mr. Paige	3773	21
Cross Examination by Mr. Jacobs	3814	21
Redirect Examination by Mr. Paige	3830	21
Recross Examination by Mr. Jacobs	3834	21
AUGUST, DAVID		
(SWORN)	3835	21
Direct Examination by Mr. Kamber	3836	21

- - - -

E X H I B I T S

TRIAL EXHIBITS	IDEN	VOL.	EVID	VOL.
2672			3776	21
3541	3862	21		
3542	3862	21		
3543	3814	21		

- - -

CERTIFICATE OF REPORTERS

We, KATHERINE POWELL SULLIVAN and DEBRA L. PAS, Official Reporters for the United States Court, Northern District of California, hereby certify that the foregoing proceedings in C 10-3561 WHA, **Oracle America, Inc., vs. Google, Inc.,** were reported by us, certified shorthand reporters, and were thereafter transcribed under our direction into typewriting; that the foregoing is a full, complete and true record of said proceedings at the time of filing.

/s/ Katherine Powell Sullivan

Katherine Powell Sullivan, CSR #5812, RPR, CRR
U.S. Court Reporter

/s/ Debra L. Pas

Debra L. Pas, CSR #11916, RMR CRR

Friday, May 11, 2012